

LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI

LI
LI

L
L
L
L
L
L
L
L
L
N
N
O
O

LO
LO
LO
LO
MA

MO
MO
MO
MO
MO
MO
MO

MC
MC

73

69

69
62
20

69
74
6C

```
1 0001 0 Module REBUILD (addressing_mode(external=GENERAL,  
2 0002 0 nonexternal=LONG_RELATIVE),  
3 0003 0 language(Bliss32),  
4 0004 0 ident = 'V04-000') =  
5 0005 1 begin  
6 0006 1  
7 0007 1 *****  
8 0008 1 *  
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *  
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *  
11 0011 1 * ALL RIGHTS RESERVED. *  
12 0012 1 *  
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *  
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *  
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *  
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *  
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *  
18 0018 1 * TRANSFERRED. *  
19 0019 1 *  
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *  
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *  
22 0022 1 * CORPORATION. *  
23 0023 1 *  
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *  
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *  
26 0026 1 *  
27 0027 1 *  
28 0028 1 *****  
29 0029 1  
30 0030 1 ++  
31 0031 1  
32 0032 1 FACILITY: VMS Mount Utility  
33 0033 1  
34 0034 1 ABSTRACT:  
35 0035 1  
36 0036 1 This module implements the rebuilding of the index file bitmap,  
37 0037 1 allocation bitmap and the quota file when a volume is mounted.  
38 0038 1  
39 0039 1 ENVIRONMENT:  
40 0040 1  
41 0041 1 VAX/VMS Operating System  
42 0042 1  
43 0043 1 --  
44 0044 1  
45 0045 1  
46 0046 1 AUTHOR: Richard I. Hustvedt, CREATION DATE: 28-Dec-1979  
47 0047 1 Andrew C. Goldstein  
48 0048 1  
49 0049 1 MODIFIED BY:  
50 0050 1  
51 0051 1 V03-016 CDS0006 Christian D. Saether 4-Sep-1984  
52 0052 1 Don't try to use the same FIB that everyone else uses  
53 0053 1 when reading the SCB.  
54 0054 1  
55 0055 1 V03-015 CDS0005 Christian D. Saether 28-Aug-1984  
56 0056 1 Define additional bit in BUILD_FLAGS argument to  
57 0057 1 REBUILD to allow REBUILD to determine whether rebuild
```

69
74

6C

69
62

20

69
60

57

69
20

55

REBUILD
V04-000

B 5
16-Sep-1984 01:27:55 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:34 [MOUNT.SRC]REBUILD.B32;2

Page 2
(1)

```

58      0058 1      is really necessary or not.
59      0059 1      Always unlock volume set when exiting for any reason.
60      0060 1
61      0061 1      V03-014 LMP0305      L. Mark Pilant,      23-Aug-1984 12:42
62      0062 1      Fix a bug that caused COUNT_QUOTA to ACCVIO if the 'UIC' was
63      0063 1      an identifier (i.e., negative).
64      0064 1
65      0065 1      V03-013 JRL0031      John R. Lawson, Jr.      24-Jul-1984 10:46
66      0066 1      Improve performance of QUOTA rebuild using chained hash table
67      0067 1
68      0068 1      V03-012 CDS0004      Christian D. Saether      10-July-1984
69      0069 1      Add STAND_ALONE_REBUILD routine to jacket the
70      0070 1      REBUILD routine when not called from MOUNT.
71      0071 1
72      0072 1      V03-011 CDS0003      Christian D. Saether      8-Dec-1983
73      0073 1      Use MOUDEF.B32 instead of FCPDEF.B32.
74      0074 1
75      0075 1      V03-010 CDS0002      Christian D. Saether      18-Oct-1983
76      0076 1      Stop clearing SCBSW_WRITECNT. This is done in MOUNT.
77      0077 1
78      0078 1      V03-009 STJ3112      Steven T. Jeffreys,      21-Jul-1983
79      0079 1      Stop-gap fix to UPDATE_ALLOCMAP to prevent ACCVIO
80      0080 1      when called from USER mode.
81      0081 1
82      0082 1      V03-008 STJ3110      Steven T. Jeffreys,      17-Jul-1983
83      0083 1      Fix bug introduced in STJ3084.
84      0084 1
85      0085 1      V03-007 TCM0001      Trudy C. Matthews      10-Jun-1983
86      0086 1      Add new input parameter (passed in R4) to call to
87      0087 1      IOC$CVT_DEVNAM.
88      0088 1
89      0089 1      V03-006 STJ3084      Steven T. Jeffreys,      31-Mar-1983
90      0090 1      Erase blocks returned to the storage bitmap.
91      0091 1
92      0092 1      V03-005 ACG0325      Andrew C. Goldstein,      4-Apr-1983 14:00
93      0093 1      Change use of file header area length symbol
94      0094 1
95      0095 1      V03-004 CDS0001      Christian D. Saether      12-Jan-1983
96      0096 1      Clear SCBSW_WRITECNT when allocation flags are cleared.
97      0097 1
98      0098 1      V03-003 STJ262      Steven T. Jeffreys,      23-Apr-1982
99      0099 1      Do cleanup before signaling secondary error condition.
100     0100 1      Change status of condition from ERROR to WARNING.
101     0101 1
102     0102 1      V03-002 STJ0254      Steven T. Jeffreys,      04-Apr-1982
103     0103 1      Use common !/O routines where possible. Duplicate
104     0104 1      the necessary macros from MOUDEF.B32.
105     0105 1
106     0106 1      V03-001 ACG0273      Andrew C. Goldstein,      26-Mar-1982 15:59
107     0107 1      Use random file sequence number in bad file headers
108     0108 1
109     0109 1      V02-022 STJ0219      Steven T. Jeffreys,      16-Feb-1982
110     0110 1      Cancel exit handler before declaring it, to ensure that
111     0111 1      a duplicate entry is not declared.
112     0112 1
113     0113 1      V02-021 BLS0147      Benn Schreiber      12-Feb-1982
114     0114 1      Reference locals with LONG_RELATIVE
```

RE
V04

69
20

55

65
60

20
72

20
61

68
20

REBUILD
V04-000

C 5
16-Sep-1984 01:27:55 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:34 [MOUNT.SRC]REBUILD.B32;2

Page 3
(1)

REB
V04

```

115 0115 1
116 0116 1 V02-020 STJ0197 Steven T. Jeffreys, 02-Feb-1982
117 0117 1 Made all external references use general addressing mode.
118 0118 1
119 0119 1 V02-019 ACG0255 Andrew C. Goldstein, 19-Jan-1982 18:28
120 0120 1 Use dynamic storage for volume descriptor tables
121 0121 1
122 0122 1 V02-018 STJ0184 Steven T. Jeffreys, 13-Jan-1982
123 0123 1 Made all references to the exit handler internal,
124 0124 1 so that the caller need have no knowledge of it.
125 0125 1
126 0126 1 V02-017 STJ0178 Steven T. Jeffreys, 04-Jan-1982
127 0127 1 Made EXIT_HNDLR_DESC a global symbol.
128 0128 1
129 0129 1 V02-016 ACG0234 Andrew C. Goldstein, 4-Dec-1981 17:11
130 0130 1 Fix EOF handling on index file bitmap
131 0131 1
132 0132 1 V02-015 MLJ0060 Martin L. Jack, 10-Nov-1981 21:23
133 0133 1 Correct V02-014 to include all rewrites of SCB.
134 0134 1
135 0135 1 V02-014 MLJ0027 Martin L. Jack, 2-Jul-1981 10:01
136 0136 1 Checksum storage control block before rewriting it.
137 0137 1
138 0138 1 V02-013 STJ0056 Steven T. Jeffreys, 29-Jun-1981
139 0139 1 Change external references to use general addressing mode.
140 0140 1
141 0141 1 V02-012 ACG0198 Andrew C. Goldstein, 5-Mar-1981 23:21
142 0142 1 Bounds check the LBN in storage map rebuild
143 0143 1
144 0144 1 V02-010 ACG0181 Andrew C. Goldstein, 9-Oct-1980 15:47
145 0145 1 Fix cross facility source reference
146 0146 1
147 0147 1 V02-009 ACG0167 Andrew C. Goldstein, 18-Apr-1980 13:39
148 0148 1 Previous revision history moved to MOUNT.REV
149 0149 1 **
150 0150 1
151 0151 1
152 0152 1 Library
153 0153 1
154 0154 1 'SYSS$LIBRARY:LIB.L32';
155 0155 1
156 0156 1 require
157 0157 1
158 0158 1 'SRC$:MOUDEF.B32';
159 0690 1
160 0691 1 LINKAGE
161 0692 1 L_ONE_ARG_OUT = CALL ( ; REGISTER = 1);
162 0693 1
163 0694 1 forward routine
164 0695 1
165 0696 1 REBUILD, ! REBUILD main routine
166 0697 1 CHECK_SCB_STATUS : L_ONE_ARG_OUT NOVALUE, ! Check dirty flags in SCB.
167 0698 1 GET_EOF, ! Get end of file on index file
168 0699 1 ALLOCATE_TABLE: novalue, ! Allocate the usage table
169 0700 1 COUNT_QUOTA: novalue, ! Count blocks in quota list
170 0701 1 DELETE_TABLE: novalue, ! Delete the quota list
171 0702 1 MARK_ALLOC: novalue, ! Mark blocks in use in bitmap
```

65
76
20

```
: 172      0703 1  VERIFY_HEADER,      : Validate file header
: 173      0704 1  READ_HOMEBLOCK: novalue,      : Read home block on volume
: 174      0705 1  CHECK_HOMEBLOCK,      : Validate home block
: 175      0706 1  RBLD_HANDLER,      : Facility condition handler
: 176      0707 1  RBLD_EXIT_HNDL: novalue,      : Facility exit handler
: 177      0708 1  FILE_SIZE,      : Compute filesize
: 178      0709 1  UPDATE_ALLOCMAP,      : Write new BITMAP.SYS
: 179      0710 1  SET_FREE: novalue;      : Set number of free blocks in VCB
: 180      0711 1
: 181      0712 1  structure
: 182      0713 1
: 183      0714 1  EXIT_CTRL_BLK [I, N] =      : Exit handler descriptor
: 184      0715 1  [74+N)*4]      : N = # of arguments ( N <= 1)
: 185      0716 1  (EXIT_CTRL_BLK+I*4)<0,32,0>;      : The block is a longword array
: 186      0717 1
: 187      0718 1  :
: 188      0719 1  Macro to signal error exit.
: 189      0720 1  :
: 190      0721 1
: 191      0722 1  macro
: 192      0723 1
: 193      0724 1  RBLD_EXIT[] = signal_stop(%REMAINING) %;
: 194      0725 1
: 195      0726 1  :
: 196      0727 1  Macro to signal error message.
: 197      0728 1  :
: 198      0729 1
: 199      0730 1  macro
: 200      0731 1
: 201      0732 1  RBLD_MESSAGE[] = signal(%REMAINING) %;
: 202      0733 1
: 203      0734 1  :+
: 204      0735 1
: 205      0736 1  Error messages
: 206      0737 1  :
: 207      0738 1  Macro to generate each error message.
: 208      0739 1  :
: 209      0740 1  :-
: 210      0741 1
: 211      0742 1  macro
: 212      0743 1
: 213      M 0744 1  ERR_TEXT(CODE, COUNT, SEVERITY, STRING) =
: 214      M 0745 1  [literal
: 215      M 0746 1  %name('RBLDS_', CODE) = MSG_CODE + FAC_CODE*16;
: 216      M 0747 1  switches
: 217      M 0748 1  UNAMES;
: 218      M 0749 1  psect
: 219      M 0750 1  own = $MSG_TEXT;
: 220      M 0751 1  own
: 221      M 0752 1  MSG_TEXT: vector[%charcount(CODE)+11+%charcount(STRING)+2, byte]
: 222      M 0753 1  initial(byte(COUNT, %charcount(CODE)+11+%charcount(STRING),
: 223      M 0754 1  %string(STRING), %string(SEVERITY), '- ',
: 224      M 0755 1  %string(CODE), ' ', STRING));
: 225      M 0756 1  psect
: 226      M 0757 1  own = $MSG_INDEX;
: 227      M 0758 1  own
: 228      M 0759 1  MSG_INDEX: initial(MSG_TEXT);
```

```
229 M 0760 1 undeclare
230 M 0761 1 MSG_TEXT,
231 M 0762 1 MSG_INDEX;
232 M 0763 1 switches
233 M 0764 1 NOUNAMES;
234 M 0765 1 %assign(MSG_CODE, MSG_CODE+8)
235 M 0766 1 psect
236 0767 1 own = $OWNS; %;
237 0768 1
238 0769 1 !
239 0770 1 ! Initialize and label the message sections.
240 0771 1 !
241 0772 1
242 0773 1 psect
243 0774 1
244 0775 1 OWN = $MSG_TEXT(nowrite, align(0));
245 0776 1
246 0777 1 own
247 0778 1
248 0779 1 MESSAGE_TEXT: vector[0, byte];
249 0780 1
250 0781 1 psect
251 0782 1
252 0783 1 OWN = $MSG_INDEX(nowrite, align(2));
253 0784 1
254 0785 1 own
255 0786 1
256 0787 1 MESSAGE_TABLE: vector[0];
257 0788 1
258 0789 1 compiletime
259 0790 1
260 0791 1 MSG_CODE = 0;
261 0792 1
262 0793 1 !
263 0794 1 ! Generate the error messages
264 0795 1 !
265 0796 1
266 0797 1 literal
267 0798 1
268 0799 1 FAC_CODE = 69; ! Or whatever
269 0800 1
270 0801 1 ERR_TEXT(NODEVICE, 0, E, 'no device currently selected');
271 0802 1 ERR_TEXT(ADDERR, 0, E, 'error adding entry');
272 0803 1 ERR_TEXT(MODIFYERR, 0, E, 'error modifying quota file');
273 0804 1 ERR_TEXT(CLOSERR, 0, E, 'error closing quota file');
274 0805 1 ERR_TEXT(LOCKERR, 0, E, 'failed to lock volume');
275 0806 1 ERR_TEXT(UNLOCKERR, 0, E, 'failed to unlock volume');
276 0807 1 ERR_TEXT(MAXVOLS, 0, E, 'volume set has too many volumes to handle');
277 0808 1 ERR_TEXT(DUALLOC, 1, W, 'dual allocation on volume !UW');
278 0809 1 ERR_TEXT(ACCINDEXF, 1, E, 'failed to access index file on relative volume !UW');
279 0810 1 ERR_TEXT(ACCBITMAP, 1, E, 'failed to access bitmap file on relative volume !UW');
280 0811 1 ERR_TEXT(ACCQFILE, 0, E, 'failed to access quota file');
281 0812 1 ERR_TEXT(QUOTARERR, 0, E, 'I/O error reading quota file');
282 0813 1 ERR_TEXT(BITMAPERR, 1, E, 'I/O error reading index file bitmap on relative volume !UW');
283 0814 1 ERR_TEXT(READSCB, 1, E, 'I/O error reading storage control block on relative volume !UW');
284 0815 1 ERR_TEXT(WRITESCB, 1, E, 'I/O error writing storage control block on relative volume !UW');
285 0816 1 ERR_TEXT(WRTIBMAP, 1, E, 'I/O error writing index file bitmap on relative volume !UW');
```

```
286 0817 1 ERR_TEXT(WRTBITMAP, 1, E, 'I/O error writing storage bitmap on relative volume !UW');
287 0818 1 ERR_TEXT(HEADERERR, 2, W, 'I/O error reading file header !UL on relative volume !UW');
288 0819 1 ERR_TEXT(WRITEHDR, 2, W, 'I/O error writing file header !UL on relative volume !UW');
289 0820 1 ERR_TEXT(MEMALLOC, 0, E, 'cannot allocate sufficient memory');
290 0821 1 ERR_TEXT(HOMEBLOCK, 1, E, 'failed to read home block on relative volume !UW');
291 0822 1 ERR_TEXT(SYSHEADER, 0, E, 'failed to read system file header - rebuild aborted');
292 0823 1 ERR_TEXT(ERRORS, 0, E, 'too many file header errors - rebuild aborted');
293 0824 1 ERR_TEXT(ERASEBLKS, 1, W, 'blocks reclaimed on relative volume !UW not completely erased');
294 0825 1
295 0826 1
296 0827 1      Module own storage.
297 0828 1
298 0829 1
299 0830 1 literal
300 0831 1
301 0832 1      MAX VOLUMES = 255,                      ! Largest volume set handled
302 0833 1      COMMAND_LENGTH = 132,
303 0834 1      OUTPUT_LENGTH = 132,
304 0835 1      BLOCK_FACTOR = 64,                      ! Blocking factor to read index file
305 0836 1
306 0837 1
307 0838 1      The following are indexes into the Exit Handler Control Block
308 0839 1
309 0840 1
310 0841 1      XHNDLR_ADDRESS = 1,                      ! Exit handler address
311 0842 1      XHNDLR_ARGCNT = 2,                      ! Exit handler argument count
312 0843 1      XHNDLR_STSADDR = 3;                     ! System exit status address
313 0844 1
314 0845 1 own
315 0846 1
316 0847 1      OWN_START: vector[0],                      ! Start of own storage
317 0848 1      BUFFER: ref BBLOCK,                      ! I/O buffer to read everything
318 0849 1      IFILEMAP: ref bitvector,                 ! Pointer to index file bitmap
319 0850 1      IFILEMAP_SIZE,                          ! Size of index file bitmap
320 0851 1      ALLOCMAP: ref bitvector,                 ! Pointer to allocation bitmap
321 0852 1      ALLOCMAP_SIZE,                          ! Size of allocation bitmap in bytes
322 0853 1      ALLOC_CLUSTER,                          ! Blocks per cluster
323 0854 1      BLOCKS_AVAIL,                          ! Available blocks on volume
324 0855 1      OLD_ALLOCMAP: ref bitvector,           ! One block window into old BITMAP.SYS
325 0856 1      ERASE_CHANNEL: word,                   ! Channel for erase I/O activity
326 0857 1      CHANNEL: word,                        ! Channel for disk I/O
327 0858 1      DUALLOC,                                ! Dual allocation flag
328 0859 1      IO_STATUS: vector[4, word],             ! I/O status block
329 0860 1      OUTPUT_LINE: vector[OUTPUT_LENGTH, byte], ! Output line buffer
330 0861 1
331 0862 1
332 0863 1      OUTPUT_DESC: vector[2],                    ! Command line descriptor
333 0864 1      CLEANUP_FLAGS: bitvector[32],           ! Output line descriptor
334 0865 1      EXIT_HNDLR_DESC: EXIT_CTRL_BLK[1];      ! Exit handler descriptor
335 0866 1
336 0867 1 literal
337 0868 1
338 0869 1      CLF_UNLOCK = 0,                            ! Unlock volume set
339 0870 1      CLF_EXIT = 1,                            ! Exit command entered
340 0871 1
341 0872 1      QF_ACTIVE = 0,                             ! Quota file active
342 0873 1      COND_REBLD = 1,                          ! Conditional rebuild
```

```
343 0874 1
344 0875 1   BITMAPS = 0,
345 0876 1   QUOTAS = 1;
346 0877 1
347 0878 1
348 0879 1
349 0880 1   Quota file record buffers
350 0881 1
351 0882 1
352 0883 1 own
353 0884 1
354 0885 1   SRC_REC: BBLOCK[DQF$C_LENGTH],
355 0886 1
356 0887 1
357 0888 1   FIB for quota file operations
358 0889 1
359 0890 1
360 0891 1   QUOTA_FIB: BBLOCK[FIB$C_LENGTH],
361 0892 1
362 0893 1   DYN_SIZE,
363 0894 1   VOLUME_PRESENT: ref bitvector,
364 0895 1   CLUSTER_FACTOR: ref vector[, word],
365 0896 1   HEADER_OFFSET: ref vector[, word],
366 0897 1   BITMAP_OFFSET: ref vector[, word],
367 0898 1   EOF: ref vector,
368 0899 1   OWN_END: vector[0];
369 0900 1
370 0901 1
371 0902 1   Usage table:
372 0903 1
373 0904 1   This table consists of a single chained hash table which is
374 0905 1   allocated and initialized by a call to ALLOCATE_TABLE
375 0906 1
376 0907 1
377 0908 1 own
378 0909 1
379 0910 1   TABLE_SIZE: initial(0),
380 0911 1   ENTRIES_IN_TABLE,
381 0912 1   USAGE_TABLE: ref blockvector[, 4];
382 0913 1
383 0914 1 macro
384 0915 1
385 0916 1   UTB_L_UIC = 0, 0, 32, 0 %;
386 0917 1   UTB_L_USAGE = 1, 0, 32, 0 %;
387 0918 1   UTB_A_NEXT = 2, 0, 32, 0 %;
388 0919 1   UTB_V_PRESCAN = 3, 0, 1, 0 %;
389 0920 1   UTB_V_INUSE = 3, 1, 1, 0 %;
390 0921 1
391 0922 1
392 0923 1   Quota record descriptors
393 0924 1
394 0925 1
395 0926 1 own
396 0927 1
397 0928 1   SRC_REC_DESC: vector[2] initial(DQF$C_LENGTH, SRC_REC),
398 0929 1   QFIB_DESC: vector[2] initial(FIB$C_LENGTH, QUOTA_FIB);
399 0930 1
```

```
! used with the NEED_REBLD flags
! same here
```

```
! Size of dynamic memory for below
! Volume present flags
! Cluster factor of volumes
! VBN offset of file headers
! VBN offset of index file bitmap
! End of index file
! End of own storage to zero
```

```
! Hash key into table
! Block usage for UIC
! Link to next in chain
! Flags entry as pre-entered in QUOTA
! Flags entry as in use
```

```
: 400      0931 1 psect
: 401      0932 1
: 402      0933 1      plit = SOWNS;
: 403      0934 1
: 404      0935 1 bind
: 405      0936 1
: 406      0937 1      QFILE_NAME = DESCRIPTOR('QUOTA.SYS;1');      ! Quota file name
: 407      0938 1
: 408      0939 1 psect
: 409      0940 1
: 410      0941 1      plit = SPLITS;
: 411      0942 1
: 412      0943 1 own
: 413      0944 1
: 414      0945 1      REC_ATTR: BBLOCK[ATR$$RECATTR],      ! Record attributes buffer
: 415      0946 1      RECATTR_DESC: vector[3]      ! Record attributes descriptor
: 416      0947 1      initial(word(ATR$$RECATTR, ATR$$RECATTR), REC_ATTR, 0);
: 417      0948 1
: 418      0949 1 bind
: 419      0950 1
: 420      0951 1      QUOTA_EOF = REC_ATTR[FAT$$_EFBLK];
: 421      0952 1
```

```
423 0953 1 GLOBAL ROUTINE STAND_ALONE_REBUILD (CHANNEL) =
424 0954 1
425 0955 1 ++
426 0956 1
427 0957 1 Functional Description:
428 0958 1
429 0959 1 This routine is a jacket routine for the main REBUILD routine.
430 0960 1 It checks to see if quotas are enabled and sets the BUILD_FLAGS
431 0961 1 argument for the REBUILD routine appropriately.
432 0962 1 This routine is used to do a rebuild any time after the disk
433 0963 1 is mounted.
434 0964 1
435 0965 1 Calling sequence:
436 0966 1 standard
437 0967 1
438 0968 1 Inputs:
439 0969 1 CHANNEL - a channel assigned to the disk to be rebuilt
440 0970 1
441 0971 1 Outputs, side effects, etc:
442 0972 1 See REBUILD.
443 0973 1
444 0974 1 --
445 0975 1
446 0976 2 BEGIN
447 0977 2
448 0978 2 LOCAL
449 0979 2 RBLD_FLAGS : BITVECTOR [2],
450 0980 2 STATUS;
451 0981 2
452 0982 2 RBLD_FLAGS [QF_ACTIVE] = 0;
453 0983 2
454 0984 2 ! All we want to find out is if the quota checking is enabled or not.
455 0985 2 ! The slimy test that follows is using the carnal knowledge that
456 0986 2 ! the check for whether quota checking is turned on will be made
457 0987 2 ! by the file system before it gets around to noticing that most
458 0988 2 ! of the required arguments are missing.
459 0989 2
460 0990 2
461 0991 2 QUOTA_FIB [FIBSW_CNTRLFUNC] = FIBSC_EXA_QUOTA;
462 0992 2
463 P 0993 2 STATUS = DO_IO (CHAN = .CHANNEL,
464 P 0994 2 FUNC = IOS_ACPCONTROL,
465 P 0995 2 IOSB = IO_STATUS,
466 0996 2 P1 = QFIB_DESC);
467 0997 2
468 0998 2 IF .STATUS AND .IO_STATUS [0] NEQ SS$_QFNOTACT
469 0999 2 THEN
470 1000 2 RBLD_FLAGS [QF_ACTIVE] = 1;
471 1001 2
472 1002 2 ! This flag tells REBUILD to only do it if the status flags
473 1003 2 ! in the storage control block indicate it is still necessary.
474 1004 2
475 1005 2
476 1006 2 RBLD_FLAGS [COND_REBLD] = 1;
477 1007 2
478 1008 2 REBUILD (.CHANNEL, .RBLD_FLAGS)
479 1009 2
```

: 480

```

J 5
16-Sep-1984 01:27:55 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:34 [MOUNT.SRC]REBUILD.B32;2

```

Page 10
(2)

REE
V04

```
.PSECT MSG_INDEX,NOWRT,NOEXE,2
```

```

000000 MESSAGE_TABLE:
000000000' 000000 :MSG_INDEX
000000000' 000004 U.2: ADDRESS U.1
000000000' 000004 :MSG_INDEX
000000000' 000008 U.4: ADDRESS U.3
000000000' 000008 :MSG_INDEX
000000000' 00000C U.6: ADDRESS U.5
000000000' 00000C :MSG_INDEX
000000000' 000010 U.8: ADDRESS U.7
000000000' 000010 :MSG_INDEX
000000000' 000014 U.10: ADDRESS U.9
000000000' 000014 :MSG_INDEX
000000000' 000018 U.12: ADDRESS U.11
000000000' 000018 :MSG_INDEX
000000000' 00001C U.14: ADDRESS U.13
000000000' 00001C :MSG_INDEX
000000000' 000020 U.16: ADDRESS U.15
000000000' 000020 :MSG_INDEX
000000000' 000024 U.18: ADDRESS U.17
000000000' 000024 :MSG_INDEX
000000000' 000028 U.20: ADDRESS U.19
000000000' 000028 :MSG_INDEX
000000000' 00002C U.22: ADDRESS U.21
000000000' 00002C :MSG_INDEX
000000000' 000030 U.24: ADDRESS U.23
000000000' 000030 :MSG_INDEX
000000000' 000034 U.26: ADDRESS U.25
000000000' 000034 :MSG_INDEX
000000000' 000038 U.28: ADDRESS U.27
000000000' 000038 :MSG_INDEX
000000000' 00003C U.30: ADDRESS U.29
000000000' 00003C :MSG_INDEX
000000000' 000040 U.32: ADDRESS U.31
000000000' 000040 :MSG_INDEX
000000000' 000044 U.34: ADDRESS U.33
000000000' 000044 :MSG_INDEX
000000000' 000048 U.36: ADDRESS U.35
000000000' 000048 :MSG_INDEX
000000000' 00004C U.38: ADDRESS U.37
000000000' 00004C :MSG_INDEX
000000000' 000050 U.40: ADDRESS U.39
000000000' 000050 :MSG_INDEX
000000000' 000054 U.42: ADDRESS U.41
000000000' 000054 :MSG_INDEX
000000000' 000058 U.44: ADDRESS U.43
000000000' 000058 :MSG_INDEX
000000000' 00005C U.46: ADDRESS U.45
000000000' 00005C :MSG_INDEX

```


REBUILD
V04-000

L 5
16-Sep-1984 01:27:55
14-Sep-1984 12:45:34

VAX-11 Bliss-32 V4.0-742
[MOUNT.SRC]REBUILD.B32;2

Page 12
(2)

REF
V04

Line	Address	Offset	Hex	ASCII	Comment
	000E0		.BLKB	1	
	000E1		.BLKB	3	
	000E4		:MSG_TEXT		
	U.11:		.BYTE	0, 43	
	000E6		.ASCII	\RBLD-\	
	000EC		.ASCII	\E\	
	000ED		.ASCII	\-\	
	000EE		.ASCII	\UNLOCKERR\	
	000F7		.ASCII	\, \	
	000F9		.ASCII	\failed to unlock volume\	
	00108				
	00110		.BLKB	1	
	00111		.BLKB	3	
	00114		:MSG_TEXT		
	U.13:		.BYTE	0, 59	
	00116		.ASCII	\RBLD-\	
	0011C		.ASCII	\E\	
	0011D		.ASCII	\-\	
	0011E		.ASCII	\MAXVOLS\	
	00125		.ASCII	\, \	
	00127		.ASCII	\volume set has too many volumes to handl\	
	00136				
	00145				
	0014F		.ASCII	\e\	
	00150		.BLKB	1	
	00151		.BLKB	3	
	00154		:MSG_TEXT		
	U.15:		.BYTE	1, 47	
	00156		.ASCII	\RBLD-\	
	0015C		.ASCII	\W\	
	0015D		.ASCII	\-\	
	0015E		.ASCII	\DUALLOC\	
	00165		.ASCII	\, \	
	00167		.ASCII	\dual allocation on volume !UW\	
	00176				
	00184		.BLKB	1	
	00185		.BLKB	3	
	00188		:MSG_TEXT		
	U.17:		.BYTE	1, 70	
	0018A		.ASCII	\RBLD-\	
	00190		.ASCII	\E\	
	00191		.ASCII	\-\	
	00192		.ASCII	\ACCINDEXF\	
	0019B		.ASCII	\, \	
	0019D		.ASCII	\failed to access index file on relative \	
	001AC				
	001BB				
	001C5		.ASCII	\volume !UW\	
	001CF		.BLKB	1	
	001D0		:MSG_TEXT		
	U.19:		.BYTE	1, 71	
	001D2		.ASCII	\RBLD-\	
	001D8		.ASCII	\E\	
	001D9		.ASCII	\-\	
	001DA		.ASCII	\ACCBITMAP\	
	001E3		.ASCII	\, \	
	001E5		.ASCII	\failed to access bitmap file on relative\	

REF
V04

REBUILD
V04-000

N 5
16-Sep-1984 01:27:55 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:34 [MOUNT.SRC]REBUILD.B32;2

Page 14
(2)

							42	43	53	45	54	49	52	57	0032D	.ASCII	\-\		
													20	2C	0032E	.ASCII	\WRITESC\		
69	74	69	72	77	20	72	6F	72	72	65	20	4F	2F	49	00336	.ASCII	\-\		
74	6E	6F	63	20	65	67	61	72	6F	74	73	20	67	6E	00338	.ASCII	\i/O error writing storage control block \		
					20	68	63	6F	6C	62	20	6C	6F	72	00347				
6C	6F	76	20	65	76	69	74	61	6C	65	72	20	6E	6F	00356				
							57	55	21	20	65	6D	75		00360	.ASCII	\on relative volume !UW\		
															0036F				
															00376	.BLKB	1		
															00377	.BLKB	1		
													4D	01	00378	;MSG TEXT			
															U.31:	.BYTE	1, 77		
								2D	44	4C	42	52	25		0037A	.ASCII	\RBLD-\		
													45		00380	.ASCII	\E\		
													2D		00381	.ASCII	\-\		
							50	41	4D	42	49	54	52	57	00382	.ASCII	\WRTIBMAP\		
69	74	69	72	77	20	72	6F	72	72	65	20	4F	2F	49	0038A	.ASCII	\-\		
62	20	65	6C	69	66	20	78	65	64	6E	69	20	67	6E	0038C	.ASCII	\i/O error writing index file bitmap on r\		
					72	20	6E	6F	20	70	61	6D	74	69	0039B				
20	65	6D	75	6C	6F	76	20	65	76	69	74	61	6C	65	003AA				
											57	55	21		003B4	.ASCII	\relative volume !UW\		
															003C3				
															003C6	.BLKB	1		
															003C7	.BLKB	1		
													48	01	003C8	;MSG TEXT			
															U.33:	.BYTE	1, 75		
								2D	44	4C	42	52	25		003CA	.ASCII	\RBLD-\		
													45		003D0	.ASCII	\E\		
													2D		003D1	.ASCII	\-\		
							50	41	4D	54	49	42	54	52	57	003D2	.ASCII	\WRTBITMAP\	
69	74	69	72	77	20	72	6F	72	72	65	20	4F	2F	49	003DB	.ASCII	\-\		
6D	74	69	62	20	65	67	61	72	6F	74	73	20	67	6E	003DD	.ASCII	\i/O error writing storage bitmap on rela\		
					61	6C	65	72	20	6E	6F	20	70	61	003EC				
57	55	21	20	65	6D	75	6C	6F	76	20	65	76	69	74	003FB				
															00405	.ASCII	\tive volume !UW\		
															00414	.BLKB	1		
															00415	.BLKB	3		
													4C	02	00418	;MSG TEXT			
															U.35:	.BYTE	2, 76		
								2D	44	4C	42	52	25		0041A	.ASCII	\RBLD-\		
													57		00420	.ASCII	\W\		
													2D		00421	.ASCII	\-\		
							52	52	45	52	45	44	41	45	48	00422	.ASCII	\HEADERERR\	
69	64	61	65	72	20	72	6F	72	72	65	20	4F	2F	49	00429	.ASCII	\-\		
20	72	65	64	61	65	68	20	65	6C	69	66	20	67	6E	0043C	.ASCII	\i/O error reading file header !UL on rel\		
					6C	65	72	20	6E	6F	20	4C	55	21	0044B				
55	21	20	65	6D	75	6C	6F	76	20	65	76	69	74	61	00455	.ASCII	\ative volume !UW\		
														57	00464				
															00465	.BLKB	1		
															00466	.BLKB	2		
													48	02	00468	;MSG TEXT			
															U.37:	.BYTE	2, 75		
								2D	44	4C	42	52	25		0046A	.ASCII	\RBLD-\		
													57		00470	.ASCII	\W\		
													2D		00471	.ASCII	\-\		
							52	44	48	45	54	49	52	57	00472	.ASCII	\WRITEHDR\		

REBUILD
V04-000

B 6
16-Sep-1984 01:27:55 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:34 [MOUNT.SRC]REBUILD.B32;2

Page 15
(2)

69	74	69	72	77	20	72	6F	72	72	65	20	4F	20	2C	0047A	.ASCII	\\	:
20	72	65	64	61	65	68	20	65	6C	69	66	20	67	6E	0047C	.ASCII	\\	:
					6C	65	72	20	6E	6F	20	4C	55	21	0048B		\\	:
55	21	20	65	6D	75	6C	6F	76	20	65	76	69	74	61	0049A	.ASCII	\\	:
														57	004A4		\\	:
															004B3		\\	:
															004B4	.BLKB	1	:
															004B5	.BLKB	3	:
													34	00	004B8	.MSG_TEXT		:
															U.39:	.BYTE	0, 52	:
									2D	44	4C	42	52	25	004BA	.ASCII	\\	:
														45	004C0	.ASCII	\\	:
														2D	004C1	.ASCII	\\	:
															004C2	.ASCII	\\	:
															004CA	.ASCII	\\	:
65	74	61	63	6F	6C	6C	61	20	74	6F	6E	6E	61	63	004CC	.ASCII	\\	:
6D	65	6D	20	74	6E	65	69	63	69	66	66	75	73	20	004DB		\\	:
												79	72	6F	004EA		\\	:
															004ED	.BLKB	1	:
															004EE	.BLKB	2	:
															004F0	.MSG_TEXT		:
															U.41:	.BYTE	1, 68	:
															004F2	.ASCII	\\	:
															004F8	.ASCII	\\	:
															004F9	.ASCII	\\	:
															004FA	.ASCII	\\	:
															00503	.ASCII	\\	:
20	64	61	65	72	20	6F	74	20	64	65	6C	69	61	66	00505	.ASCII	\\	:
72	20	6E	6F	20	68	63	6F	6C	62	20	65	6D	6F	68	00514		\\	:
					6F	76	20	65	76	69	74	61	6C	65	00523		\\	:
															0052D	.ASCII	\\	:
															00535	.BLKB	1	:
															00536	.BLKB	2	:
															00538	.MSG_TEXT		:
															U.43:	.BYTE	0, 71	:
															0053A	.ASCII	\\	:
															00540	.ASCII	\\	:
															00541	.ASCII	\\	:
															00542	.ASCII	\\	:
															0054B	.ASCII	\\	:
20	64	61	65	72	20	6F	74	20	64	65	6C	69	61	66	0054D	.ASCII	\\	:
61	65	68	20	65	6C	69	66	20	6D	65	74	73	79	73	0055C		\\	:
															0056B		\\	:
															00575	.ASCII	\\	:
															00580	.BLKB	1	:
															00581	.BLKB	3	:
															00584	.MSG_TEXT		:
															U.45:	.BYTE	0, 62	:
															00586	.ASCII	\\	:
															0058C	.ASCII	\\	:
															0058D	.ASCII	\\	:
															0058E	.ASCII	\\	:
															00594	.ASCII	\\	:
68	20	65	6C	69	66	20	79	6E	61	6D	20	6F	6F	74	00596	.ASCII	\\	:
20	20	20	73	72	6F	72	72	65	20	72	65	64	61	65	005A5		\\	:
															005B4		\\	:
															005BE	.ASCII	\\	:

C 6
16-Sep-1984 01:27:55
14-Sep-1984 12:45:34

Page 16
(2)

REI
VO4

65	6D	69	61	6C	63	65	72	20	73	6B	63	6F	6C	62	005D9	.ASCII	\blocks reclaimed on relative volume !UW \
76	20	65	76	69	74	61	6C	65	72	20	6E	6F	20	64	005E8		
20	79	6C	65	74	65	6C	70	6D	6F	63	20	74	6F	6E	00601	.ASCII	\not completely erased\
									64	65	73	61	72	65	00610	.BLKB	1
															00616	.PSECT	\$OWNS,NOEXE,2
															00000	OWN_START:	
																.BLKB	0
															00000	BUFFER:	.BLKB 4
															00004	IFILEMAP:	.BLKB 4
																.BLKB	4
															00008	IFILEMAP_SIZE:	.BLKB 4
																.BLKB	4
															0000C	ALLOCMAP:	.BLKB 4
																.BLKB	4
															00010	ALLOCMAP_SIZE:	.BLKB 4
																.BLKB	4
															00014	ALLOC_CLUSTER:	.BLKB 4
																.BLKB	4
															00018	BLOCKS_AVAIL:	.BLKB 4
																.BLKB	4
															0001C	OLD_ALLOCMAP:	.BLKB 4
																.BLKB	4
															00020	ERASE_CHANNEL:	.BLKB 2
																.BLKB	2
															00022	CHANNEL:	.BLKB 4
															00024	DUALLOC:	.BLKB 4
															00028	IO_STATUS:	.BLKB 8
																.BLKB	132
															00030	OUTPUT_LINE:	.BLKB 8
																.BLKB	4
															000B4	OUTPUT_DESC:	.BLKB 4
																.BLKB	20
															000BC	CLEANUP_FLAGS:	.BLKB 32
																.BLKB	64
															000C0	EXIT_HNDLR_DESC:	.BLKB 4
																.BLKB	4
															000D4	SRC_REC:	.BLKB 4
															000F4	QUOTA_FIB:	.BLKB 4
																.BLKB	4
															00134	DYN_SIZE:	.BLKB 4
																.BLKB	4
															00138	VOLUME_PRESENT:	.BLKB 4
																.BLKB	4
															0013C	CLUSTER_FACTOR:	.BLKB 4

```
00140 HEADER_OFFSET:
00144 BITMAP_OFFSET:
00148 EOF:
0014C OWN_END:
00000000 0014C TABLE_SIZE:
00150 ENTRIES_IN_TABLE:
00154 USAGE_TABLE:
00000020 00158 SRCREC_DESC:
00000000' 0015C
00000040 00160 QFIB_DESC:
00000000' 00164
31 3B 53 59 53 2E 41 54 4F 55 51 00168 P.AAB:
00173
0000000B 00174 P.AAA:
00000000' 00178
0017C REC_ATTR:
0004 0020 0019C RECATTR_DESC:
00000000' 001A0
00000000 001A4
QFILE_NAME=
QUOTA_EOF=
P.AAA
REC_ATTR+8
.EXTRN COMMON_IO
.PSECT $CODE$,NOWRT,2
.ENTRY STAND ALONE REBUILD, Save R2,R3
MOVAB IO_STATUS, R3
BICB2 #1, RBLD_FLAGS
MOVW #12, QUOTA_FIB+22
CLRQ -(SP)
CLRQ -(SP)
CLRL -(SP)
PUSHAB QFIB_DESC
CLRQ -(SP)
PUSHL R3
PUSHL #56
PUSHL CHANNEL
PUSHL #26
CALLS #12, COMMON_IO
BLBC STATUS, 1$
CMPW IO_STATUS, #980
BEQL 1$
BISB2 #1, RBLD_FLAGS
BISB2 #2, RBLD_FLAGS
MOVZBL RBLD_FLAGS, -(SP)
PUSHL CHANNEL
CALLS #2, REBUILD
```

00E2 53 00000000' EF 9E 00002
52 01 8A 00009
C3 0C 80 0000C
7E 7C 00011
7E 7C 00013
7E D4 00015
0138 C3 9F 00017
7E 7C 0001B
53 DD 0001D
38 DD 0001F
04 AC DD 00021
1A DD 00024
00000000G 00 0C FB 00026
0A 50 E9 0002D
03D4 8F 63 B1 00030
52 03 13 00035
52 01 88 00037
7E 02 88 0003A 1\$:
52 9A 0003D
04 AC DD 00040
00000000V EF 02 FB 00043

0953
0982
0991
0996
0998
1000
1006
1008

RET

: 1010

```

: Routine Size: 75 bytes,      Routine Base: $CODE$ + 0000

```

```

: 482      1011 1 GLOBAL ROUTINE REBUILD (CHANNEL_ARG, BUILD_FLAGS) =
: 483      1012 1
: 484      1013 1 +-
: 485      1014 1
: 486      1015 1 Functional Description:
: 487      1016 1
: 488      1017 1 This routine implements the REBUILD function on volume mount. It scans
: 489      1018 1 the index file of each volume in the volume set and constructs a table
: 490      1019 1 of UIC's and blocks used. It then updates the usage data in the quota
: 491      1020 1 file, creating entries as needed so that all UIC's using blocks are
: 492      1021 1 listed. As the index file is scanned, the index file bitmap and
: 493      1022 1 allocation bitmaps are also rebuilt and will be rewritten.
: 494      1023 1
: 495      1024 1 Calling Sequence:
: 496      1025 1 standard
: 497      1026 1
: 498      1027 1 Input Parameters:
: 499      1028 1 CHANNEL_ARG: channel number assigned to the volume (set)
: 500      1029 1 BUILD_FLAGS:
: 501      1030 1 BIT 0 [QF_ACTIVE] : 1 to rebuild quota file, 0 to not
: 502      1031 1 BIT 1 [COND_REBLD] : 0 for unconditional rebuild, test SCB flags otherwise
: 503      1032 1
: 504      1033 1 Implicit Inputs:
: 505      1034 1 none
: 506      1035 1
: 507      1036 1 Output Parameters:
: 508      1037 1 none
: 509      1038 1
: 510      1039 1 Implicit Outputs:
: 511      1040 1 none
: 512      1041 1
: 513      1042 1 Routines Called:
: 514      1043 1 none
: 515      1044 1
: 516      1045 1 Routine Value:
: 517      1046 1 none
: 518      1047 1
: 519      1048 1 Signals:
: 520      1049 1 none
: 521      1050 1
: 522      1051 1 Side Effects:
: 523      1052 1 none
: 524      1053 1
: 525      1054 1 --
: 526      1055 1
: 527      1056 2 BEGIN
: 528      1057 2
: 529      1058 2 MAP
: 530      1059 2 BUILD_FLAGS : BITVECTOR;
: 531      1060 2
: 532      1061 2 BUILTIN
: 533      1062 2 ROT;
: 534      1063 2
: 535      1064 2 LOCAL
: 536      1065 2 STATUS, : general status value
: 537      1066 2 READ_STATUS, : status of file header read I/O
: 538      1067 2 READ_LENGTH, : number of blocks to read
```

```
539 1068 2 BLOCKS_READ,      ! number of blocks actually read
540 1069 2 ERR_COUNT,    ! count of errors encountered
541 1070 2 RETRY_COUNT,  ! number of blocks to retry in single block mode
542 1071 2 J,            ! loop index
543 1072 2 VOLUME_COUNT, ! number of volumes in set
544 1073 2 DEFAULT_QUOTA: initial(1000), ! default quota for new entries
545 1074 2 DEFAULT_OVER: initial(100),   ! default overdraft limit for new entries
546 1075 2
547 1076 2 BITNUMBER,    ! general bit pointer
548 1077 2 VBN,          ! current VBN in file
549 1078 2 BUFPTR,       ! current buffer pointer for file
550 1079 2 ENTRY         ! pointer to quota file entry
551 1080 2 : REF BBLOCK, ! pointer to file header
552 1081 2 : REF BBLOCK, ! pointer to file header
553 1082 2 : BBLOCK [FIDSC_LENGTH], ! file ID block
554 1083 2 : VECTOR [2], ! buffer for time of day
555 1084 2 : BITVECTOR [2], ! flag to indicate quota file rebuild
556 1085 2 BLOCK_COUNT, ! blocks used by header
557 1086 2 FILE_NUMBER;  ! file number of file in question
558 1087 2
559 1088 2 EXTERNAL ROUTINE
560 1089 2 CHECKSUM,      ! compute block checksum
561 1090 2 LIB$FREE_VM   ! ADDRESSING_MODE (GENERAL); ! deallocate working storage
562 1091 2 LIB$GET_VM    ! ADDRESSING_MODE (GENERAL); ! allocate working storage
563 1092 2
564 1093 2 ENABLE RBLD_HANDLER;
565 1094 2 CH$FILL (0, OWN_END-OWN_START, OWN_START);
566 1095 2 ! Set up the exit handler descriptor and declare the handler.
567 1096 2
568 1097 2
569 1098 2 EXIT_HNDLR_DESC[XHNDLR_ADDRESS] = RBLD_EXIT_HNDL;
570 1099 2 EXIT_HNDLR_DESC[XHNDLR_ARGCNT] = 1;
571 1100 2 EXIT_HNDLR_DESC[XHNDLR_STSADDR] = EXIT_HNDLR_DESC[XHNDLR_STSADDR+1];
572 1101 2
573 1102 2 $CANEXH (DESBLOCK=EXIT_HNDLR_DESC);
574 1103 2 $DCLEXH (DESBLOCK=EXIT_HNDLR_DESC);
575 1104 2
576 1105 2 ! Initialize the actual blocking factor for reads. If the working set
577 1106 2 ! turns out to be too small, decrease the blocking factor until reads succeed.
578 1107 2
579 1108 2
580 1109 2 READ_LENGTH = BLOCK_FACTOR;
581 1110 2
582 1111 2
583 1112 2 ! Verify that a channel is open.
584 1113 2
585 1114 2
586 1115 2 CHANNEL = .CHANNEL_ARG;
587 1116 2
588 1117 2 IF .CHANNEL EQL 0
589 1118 2 THEN RBLD_EXIT (RBLD$_NODEVICE);
590 1119 2
591 1120 2 ! Lock the volume set against modification.
592 1121 2
593 1122 2
594 1123 2 QUOTA_FIB[FIB$_CNTRLFUNC] = FIB$_LOCK_VOL;
595 1124 2 STATUS = DO_IO (CHAN = .CHANNEL,
```

```
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
P 1125 2      FUNC = IOS_ACPCONTROL,  
P 1126 2      IOSB = IO_STATUS,  
P 1127 2      P1  = QFIB_DESC  
1128 2      );  
1129 2      IF .STATUS THEN STATUS = .IO_STATUS[0];  
1130 2      IF NOT .STATUS  
1131 2      THEN RBLD_EXIT (RBLD$_LOCKERR, .STATUS);  
1132 2  
1133 2      !CLEANUP_FLAGS[CLF_UNLOCK] = 1;  
1134 2  
1135 2      ! Allocate the I/O buffer.  
1136 2      !  
1137 2  
1138 2      STATUS = LIB$GET_VM (UPLIT (BLOCK_FACTOR*512), BUFFER);  
1139 2      IF NOT .STATUS  
1140 2      THEN  
1141 2          BEGIN  
1142 2              BUFFER = 0;  
1143 2              RBLD_EXIT (RBLD$_MEMALLOC, 0, .STATUS);  
1144 2          END;  
1145 2  
1146 2      ! Now open the index file on RVN 1 and read the home block.  
1147 2      !  
1148 2  
1149 2      CH$FILL (0, FIB$_LENGTH, QUOTA_FIB);  
1150 2      QUOTA_FIB[FIB$_ACCTL] = FIB$_NOWRITE;  
1151 2      QUOTA_FIB[FIB$_FID_NUM] = FIB$_INDEXF;  
1152 2      QUOTA_FIB[FIB$_FID_SEQ] = FIB$_INDEXF;  
1153 2      QUOTA_FIB[FIB$_FID_RVN] = 1;  
1154 2  
1155 2      STATUS = DO_IO (CHAN = .CHANNEL,  
P 1156 2          FUNC = IOS_ACCESS OR IOSM_ACCESS,  
P 1157 2          IOSB = IO_STATUS,  
P 1158 2          P1  = QFIB_DESC  
1159 2          );  
1160 2      IF .STATUS THEN STATUS = .IO_STATUS[0];  
1161 2      IF NOT .STATUS  
1162 2      THEN RBLD_EXIT (RBLD$_ACCINDEXF, 1, .STATUS);  
1163 2  
1164 2      READ_HOMEBLOCK (.BUFFER, 1);  
1165 2  
1166 2      VOLUME_COUNT = .BUFFER[HM2$_SETCOUNT];  
1167 2      IF .VOLUME_COUNT EQL 0 THEN VOLUME_COUNT = 1;  
1168 2      IF .VOLUME_COUNT GTRU MAX_VOLUMES  
1169 2      THEN RBLD_EXIT (RBLD$_MAXVOLS);  
1170 2  
1171 2      ! Allocate the volume descriptors.  
1172 2      !  
1173 2  
1174 2      DYN_SIZE = ((.VOLUME_COUNT+7)/8) * 81;  
1175 2      STATUS = LIB$GET_VM (DYN_SIZE, EOF);  
1176 2      IF NOT .STATUS  
1177 2      THEN  
1178 2          BEGIN  
1179 2              EOF = 0;  
1180 2              RBLD_EXIT (RBLD$_MEMALLOC, 0, .STATUS);  
1181 2          END;
```

```
1182 2 CH$FILL (0, .DYN_SIZE, .EOF);
1183 2 CLUSTER_FACTOR = .EOF + .VOLUME_COUNT*4;
1184 2 HEADER_OFFSET = .CLUSTER_FACTOR + .VOLUME_COUNT*2;
1185 2 BITMAP_OFFSET = .HEADER_OFFSET + .VOLUME_COUNT*2;
1186 2 VOLUME_PRESENT = .BITMAP_OFFSET + .VOLUME_COUNT*2;
1187 2
1188 2
1189 2 VOLUME_PRESENT[0] = 1;
1190 2 CLUSTER_FACTOR[0] = .BUFFER[HM2$W_CLUSTER];
1191 2 BITMAP_OFFSET[0] = .BUFFER[HM2$W_CLUSTER]*4 + 1;
1192 2 HEADER_OFFSET[0] = .BUFFER[HM2$W_CLUSTER]*4 + .BUFFER[HM2$W_IBMAPSIZE];
1193 2 EOF[0] = GET_EOF (.BUFFER, 1);
1194 2
1195 2 DO_IO (CHAN = .CHANNEL,
1196 2       FUNC = IOS_DEACCESS
1197 2       );
1198 2
1199 2 : If this is a conditional rebuild, access the storage bitmap file to
1200 2 : read the storage control block and determine whether the various flags
1201 2 : in the status2 field indicate a rebuild is still necessary. The status2
1202 2 : flags are set when the volume is mounted if the corresponding flag was
1203 2 : set in the status flags of the SCB, and the current count of writers
1204 2 : mismatched with the current number of locks outstanding, indicating
1205 2 : that the volume had caching enabled and was improperly dismounted.
1206 2
1207 2 : If not a conditional rebuild, simply set the flags.
1208 2
1209 2
1210 2 IF .BUILD_FLAGS [COND_REBLD]
1211 2 THEN
1212 2   CHECK_SCB_STATUS (1, 0 ;NEED_REBLD)
1213 2 ELSE
1214 2   BEGIN
1215 2     NEED_REBLD [BITMAPS] = 1;
1216 2     NEED_REBLD [QUOTAS] = .BUILD_FLAGS [QF_ACTIVE];
1217 2   END;
1218 2
1219 2 : If this is a volume set, access the index file of each volume in the set
1220 2 : and get cluster factor and EOF. We do this beforehand to verify accessibility
1221 2 : of all volumes in the set.
1222 2
1223 2
1224 2 INCR J FROM 2 TO .VOLUME_COUNT
1225 2 DO
1226 2   BEGIN
1227 2     QUOTA_FIB[FIB$W_FID RVN] = .J;
1228 2     STATUS = DO_IO (CHAN = .CHANNEL,
1229 2                     FUNC = IOS_ACCESS OR IOSM_ACCESS,
1230 2                     IOSB = IO_STATUS,
1231 2                     P1 = QFIB_DESC
1232 2                     );
1233 2     IF .STATUS THEN STATUS = .IO_STATUS[0];
1234 2     IF NOT .STATUS
1235 2     THEN
1236 2       BEGIN
1237 2         RBLD_MESSAGE (RBLD$_ACCINDEXF, .J, .STATUS);
1238 2
```

```

: 710      1239  4      NEED_REBLD [QUOTAS] = 0;
: 711      1240  4      END
: 712      1241  3      ELSE
: 713      1242  4      BEGIN
: 714      1243  4      VOLUME_PRESENT[J-1] = 1;
: 715      1244  4      READ_HOMEBLOCK (.BUFFER, .J);
: 716      1245  4      CLUSTER_FACTOR[J-1] = .BUFFER[HM2$W_CLUSTER];
: 717      1246  4      BITMAP_OFFSET[J-1] = .BUFFER[HM2$W_CLUSTER]*4 + 1;
: 718      1247  4      HEADER_OFFSET[J-1] = .BUFFER[HM2$W_CLUSTER]*4 + .BUFFER[HM2$W_IBMAPSIZE];
: 719      1248  4      EOF[J-1] = GET_EOF (.BUFFER, .J);
: 720      1249  4      DO_IO (CHAN = .CHANNEL,
: 721      1250  4      FUNC = IOS_DEACCESS
: 722      1251  4      );
: 723      P 1252  4      ! If conditional rebuild, check whether it is necessary for this volume.
: 724      P 1253  4      !
: 725      1254  4      IF .BUILD_FLAGS [COND_REBLD]
: 726      1255  4      THEN
: 727      1256  4      CHECK_SCB_STATUS (.J, .NEED_REBLD ;NEED_REBLD);
: 728      1257  4      END;
: 729      1258  4      END;
: 730      1259  4      ! If this is a conditional rebuild, see if anything needs rebuilding.
: 731      1260  4      ! Don't rebuild quotas unless the quota file is active.
: 732      1261  4      ! If the quota file needs rebuilding, the bitmaps get rebuilt regardless.
: 733      1262  4      ! If nothing needs rebuilding, clean up and exit.
: 734      1263  3      IF .BUILD_FLAGS [COND_REBLD]
: 735      1264  2      THEN
: 736      1265  2      BEGIN
: 737      1266  2      IF NOT .BUILD_FLAGS [QF_ACTIVE]
: 738      1267  2      THEN
: 739      1268  2      NEED_REBLD [QUOTAS] = 0;
: 740      1269  2      IF .NEED_REBLD [QUOTAS]
: 741      1270  2      THEN
: 742      1271  2      NEED_REBLD [BITMAPS] = 1;
: 743      1272  2      IF NOT .NEED_REBLD [BITMAPS]
: 744      1273  2      THEN
: 745      1274  3      BEGIN
: 746      1275  3      QUOTA_FIB[FIB$W_CNTRLFUNC] = FIB$C_UNLK_VOL;
: 747      1276  3      QUOTA_FIB[FIB$L_CNTRLVAL] = 0;
: 748      1277  3      STATUS = DO_IO (CHAN = .CHANNEL,
: 749      1278  3      FUNC = IOS_ACPCONTROL,
: 750      1279  3      IOSB = IO_STATUS,
: 751      1280  3      P1 = QFIB_DESC
: 752      1281  3      );
: 753      1282  3      END;
: 754      1283  3      END;
: 755      1284  3      END;
: 756      1285  3      END;
: 757      1286  4      END;
: 758      1287  4      END;
: 759      1288  4      END;
: 760      1289  4      END;
: 761      P 1290  4      END;
: 762      P 1291  4      END;
: 763      P 1292  4      END;
: 764      P 1293  4      END;
: 765      1294  4      END;
: 766      1295  4      END;
```

```

: 767      1296 4      IF .STATUS THEN STATUS = .IO_STATUS[0];
: 768      1297 4
: 769      1298 4      IF NOT .STATUS
: 770      1299 4      THEN RBLD_EXIT (RBLD$_UNLOCKERR, .STATUS);
: 771      1300 4
: 772      1301 4      LIB$FREE_VM (UPLIT (BLOCK_FACTOR*512), BUFFER);
: 773      1302 4      BUFFER = 0;
: 774      1303 4      LIB$FREE_VM (DYN_SIZE, EOF);
: 775      1304 4      EOF = 0;
: 776      1305 4
: 777      1306 4      $SCANEXH (DESBLOCK=EXIT_HNDLR_DESC);
: 778      1307 4
: 779      1308 4      RETURN $$$_NORMAL
: 780      1309 3      END;
: 781      1310 3
: 782      1311 2      END;
: 783      1312 2
: 784      1313 2      ! Scan the existing quota file to prebuild the usage table. This is essential
: 785      1314 2      ! to get table entries for quota file entries that have zero usage.
: 786      1315 2      !
: 787      1316 2
: 788      1317 2      QUOTA_FIB[FIB$_ACCTL] = 0;
: 789      1318 2      QUOTA_FIB[FIB$_DID_NUM] = FIB$_MFD;
: 790      1319 2      QUOTA_FIB[FIB$_DID_SEQ] = FIB$_MFD;
: 791      1320 2      QUOTA_FIB[FIB$_DID_RVN] = 1;
: 792      1321 2      QUOTA_FIB[FIB$_CNTRLFUNC] = FIB$_ENA_QUOTA;
: 793      1322 2      IF .NEED_REBLD [QUOTAS]
: 794      1323 2      THEN
: 795      1324 3      BEGIN
: 796      1325 3      STATUS = DO_IO (CHAN = .CHANNEL,
: 797      1326 3      FUNC = IOS_ACCESS OR IOSM_ACCESS,
: 798      1327 3      IOSB = IO_STATUS,
: 799      1328 3      P1 = QFIB_DESC,
: 800      1329 3      P2 = QFILE_NAME,
: 801      1330 3      P5 = RECATTR_DESC
: 802      1331 3      );
: 803      1332 3      IF .STATUS THEN STATUS = .IO_STATUS[0];
: 804      1333 3      IF NOT .STATUS
: 805      1334 3      THEN RBLD_EXIT (RBLD$_ACQFILE, .STATUS);
: 806      1335 3
: 807      1336 3      QUOTA_EOF = ROT (.QUOTA_EOF, 16) - 1;
: 808      1337 3
: 809      1338 3      ALLOCATE_TABLE();
: 810      1339 3
: 811      1340 3      VBN = 1;
: 812      1341 3      UNTIL .VBN GTRU .QUOTA_EOF
: 813      1342 3      DO
: 814      1343 4      BEGIN
: 815      1344 4      STATUS = DO_IO (CHAN = .CHANNEL,
: 816      1345 4      FUNC = IOS_READVBLK,
: 817      1346 4      IOSB = IO_STATUS,
: 818      1347 4      P1 = BUFFER,
: 819      1348 4      P2 = $12 * MINU (.READ_LENGTH, .QUOTA_EOF - .VBN + 1),
: 820      1349 4      P3 = .VBN
: 821      1350 4      );
: 822      1351 4      IF .STATUS THEN STATUS = .IO_STATUS[0];
: 823      1352 4      IF NOT .STATUS
```

```

824      1353 4      THEN
825      1354 5      BEGIN
826      1355 5      IF .STATUS EQL $$$_INFWSL
827      1356 5      THEN
828      1357 6      BEGIN
829      1358 6      READ_LENGTH = .READ_LENGTH - 1;
830      1359 6      IF .READ_LENGTH EQL 0
831      1360 6      THEN RBLD_EXIT (RBLD$_QUOTARERR, .STATUS);
832      1361 6      IO_STATUS[1] = 0;
833      1362 6      END
834      1363 5      ELSE
835      1364 5      RBLD_EXIT (RBLD$_QUOTARERR, .STATUS);
836      1365 4      END;
837      1366 4      ENTRY = .BUFFER;
838      1367 4      UNTIL .ENTRY GEQA .BUFFER + .IO_STATUS[1]
839      1368 4      DO
840      1369 4      BEGIN
841      1370 5      IF .ENTRY[DQF$V_ACTIVE]
842      1371 5      THEN
843      1372 5      BEGIN
844      1373 6      IF .ENTRY[DQF$UIC] EQL 0
845      1374 6      THEN
846      1375 6      BEGIN
847      1376 7      DEFAULT_QUOTA = .ENTRY[DQF$PERMQUOTA];
848      1377 7      DEFAULT_OVER = .ENTRY[DQF$OVERDRAFT];
849      1378 7      END
850      1379 7      ELSE
851      1380 7      BEGIN
852      1381 7      COUNT_QUOTA (.ENTRY[DQF$UIC], 0, 1);
853      1382 6      END;
854      1383 5      END;
855      1384 5      ENTRY = .ENTRY + DQF$C_LENGTH;
856      1385 4      END;
857      1386 4      VBN = .VBN + .IO_STATUS[1] / 512;
858      1387 4      END;
859      1388 4      ! end of buffer processing loop
860      1389 3      ! end of quota file reading loop
861      1390 3      DO_IO (CHAN = .CHANNEL,
862      1391 3      FUNC = IO$_DEACCESS
863      1392 3      );
864      1393 3      END;
865      1394 2      ! Now we loop for all the volumes in the set. Open the index file and start
866      1395 2      ! reading file headers.
867      1396 2      CH$FILL (0, FIB$C_LENGTH, QUOTA_FIB);
868      1397 2      INCR J FROM 1 TO .VOLUME_COUNT
869      1398 2      DO IF .VOLUME_PRESENT[J=1]
870      1399 2      THEN
871      1400 3      BEGIN
872      1401 3      DUALLOC = 0;
873      1402 3      ERR_COUNT = 0;
874      1403 3      QUOTA_FIB[FIB$L_ACCTL] = FIB$M_NOWRITE;
875      1404 3
876      1405 3
877      1406 3
878      1407 3
879      1408 3
880      1409 3
```

```
881 1410 3 QUOTA_FIB[FIB$W_FID_RVN] = .J;
882 1411 3 QUOTA_FIB[FIB$W_FID_NUM] = FID$C_BITMAP;
883 1412 3 QUOTA_FIB[FIB$W_FID_SEQ] = FID$C_BITMAP;
884 1413 3 STATUS = DO_IO (CHAN = .CHANNEL,
885 P 1414 3 FUNC = IOS_ACCESS OR IOSM_ACCESS,
886 P 1415 3 IOSB = IO_STATUS,
887 P 1416 3 P1 = QFIB_DESC
888 1417 3 );
889 1418 3 IF .STATUS THEN STATUS = .IO_STATUS[0];
890 1419 3 IF NOT .STATUS
891 1420 3 THEN RBLD_EXIT (RBLD$_ACCBITMAP, .J, .STATUS);
892 1421 3
893 P 1422 3 STATUS = DO_IO (CHAN = .CHANNEL,
894 P 1423 3 FUNC = IOS_READVBLK,
895 P 1424 3 IOSB = IO_STATUS,
896 P 1425 3 P1 = .BUFFER,
897 P 1426 3 P2 = 512,
898 P 1427 3 P3 = 1
899 1428 3 );
900 1429 3 IF .STATUS THEN STATUS = .IO_STATUS[0];
901 1430 3 IF NOT .STATUS
902 1431 3 THEN
903 1432 3 BEGIN
904 1433 3 ! CLEANUP_FLAGS[CLF_UNLOCK] = 0;
905 1434 3 RBLD_EXIT (RBLD$_READSCB, .J, .STATUS);
906 1435 3 END;
907 1436 3
908 1437 3 ALLOC_CLUSTER = .CLUSTER_FACTOR[.J-1];
909 1438 3 ALLOCMAP_SIZE = 4095 + (.BUFFER[SCB$L_VOLSIZE] + .BUFFER[SCB$W_CLUSTER] - 1)
910 1439 3 / .BUFFER[SCB$W_CLUSTER];
911 1440 3 ALLOCMAP_SIZE = 512 * (.ALLOCMAP_SIZE / 4096); ! convert to page byte count
912 1441 3 STATUS = LIB$GET_VM (ALLOCMAP_SIZE, ALLOCMAP);
913 1442 3 IF NOT .STATUS
914 1443 3 THEN
915 1444 3 BEGIN
916 1445 3 ALLOCMAP = 0;
917 1446 3 RBLD_EXIT (RBLD$_MEMALLOC, .STATUS);
918 1447 3 END;
919 1448 3
920 1449 3 Initialize allocation bitmap to show all space available
921 1450 3
922 1451 3 CH$FILL (0, .ALLOCMAP_SIZE, .ALLOCMAP);
923 1452 3 BLOCKS_AVAIL = ((.BUFFER[SCB$L_VOLSIZE] + .BUFFER[SCB$W_CLUSTER] - 1)
924 1453 3 / .BUFFER[SCB$W_CLUSTER])
925 1454 3 * .BUFFER[SCB$W_CLUSTER];
926 1455 3 INCR BITNUMBER FROM 0
927 1456 3 TO ((.BUFFER[SCB$L_VOLSIZE] + .BUFFER[SCB$W_CLUSTER] - 1)
928 1457 3 / .BUFFER[SCB$W_CLUSTER]) - 1
929 1458 3
930 1459 3 DO
931 1460 3 BEGIN
932 1461 3 ALLOCMAP[.BITNUMBER] = 1;
933 1462 3 END;
934 P 1463 3 STATUS = DO_IO (CHAN = .CHANNEL,
935 P 1464 3 FUNC = IOS_DEACCESS
936 1465 3 );
937 1466 3
```

```

938      QUOTA_FIB[FIBSL_ACCTL] = FIBSM_NOWRITE OR FIBSM_WRITE;
939      QUOTA_FIB[FIBSW_FID_NUM] = FIDSC_INDEXF;
940      QUOTA_FIB[FIBSW_FID_SEQ] = FIDSC_INDEXF;
941
942      P 1471      STATUS = DO_IO (CHAN = .CHANNEL,
943      P 1472      FUNC = IOS_ACCESS OR IOSM_ACCESS,
944      P 1473      IOSB = IO_STATUS,
945      P 1474      P1 = QFIB_DESC
946      1475      );
947      IF .STATUS THEN STATUS = .IO_STATUS[0];
948      IF NOT .STATUS
949      THEN RBLD_EXIT (RBLDS_ACCINDEXF, .J, .STATUS);
950
951      Allocate space for working copy of index file bit map
952
953      IFILEMAP_SIZE = (.HEADER_OFFSET[J-1] - .BITMAP_OFFSET[J-1] + 1) * 512;
954      STATUS = LIB$GET_VM (IFILEMAP_SIZE, IFILEMAP);
955      IF NOT .STATUS
956      THEN
957      BEGIN
958      IFILEMAP = 0;
959      RBLD_EXIT (RBLDS_MEMALLOC, .STATUS);
960      END;
961
962      Read old index file bitmap into buffer.
963
964      P 1493      STATUS = DO_IO (CHAN = .CHANNEL,
965      P 1494      FUNC = IOS_READVBLK,
966      P 1495      IOSB = IO_STATUS,
967      P 1496      P1 = .IFILEMAP,
968      P 1497      P2 = .IFILEMAP_SIZE,
969      P 1498      P3 = .BITMAP_OFFSET[J-1]
970      1499      );
971      IF .STATUS THEN STATUS = .IO_STATUS[0];
972      IF NOT .STATUS THEN RBLD_EXIT (RBLDS_BITMAPERR, .J, .STATUS);
973
974      ! Loop for all blocks in the index file. Read headers, starting with the MFD.
975      ! We read multiple blocks into a data buffer and process them one at a
976      ! time.
977
978      VBN = .HEADER_OFFSET[J-1] + 1;
979      UNTIL .VBN GTRU .EOF[J-1]
980      DO
981      BEGIN
982      READ_STATUS = DO_IO (CHAN = .CHANNEL,
983      P 1514      FUNC = IOS_READVBLK,
984      P 1515      IOSB = IO_STATUS,
985      P 1516      P1 = .BUFFER,
986      P 1517      P2 = 512 * MINU (.READ_LENGTH, .EOF[J-1] - .VBN + 1),
987      P 1518      P3 = .VBN
988      1519      );
989      IF .READ_STATUS THEN READ_STATUS = .IO_STATUS[0] ELSE IO_STATUS = 0;
990
991      ! If an I/O error occurred, go into single block mode for the scope of the
992
993
994
```

```

: 995      1524 4 ! read. We must do this since the byte count returned with an I/O error is
: 996      1525 4 ! not reliable. If we are already in single block mode, then see if the header
: 997      1526 4 ! read was marked active; only report the error if it was.
: 998      1527 4 !
: 999      1528 4 !
: 1000     1529 4      IF NOT .READ_STATUS
: 1001     1530 4      THEN
: 1002     1531 5          BEGIN
: 1003     1532 5              IO_STATUS = 0;
: 1004     1533 5              IF .READ_STATUS EQL SSS_INSFWSL
: 1005     1534 5                  THEN
: 1006     1535 6                  BEGIN
: 1007     1536 6                      READ_LENGTH = .READ_LENGTH - 1;
: 1008     1537 6                      IF .READ_LENGTH EQL 0
: 1009     1538 6                          THEN RBLD_EXIT (RBLD$_HEADERERR, .FILE_NUMBER, .J, .READ_STATUS);
: 1010     1539 6                      END
: 1011     1540 6                  ELSE
: 1012     1541 5                      BEGIN
: 1013     1542 6                          IF .READ_LENGTH GTRU 1
: 1014     1543 6                              THEN
: 1015     1544 6                                  BEGIN
: 1016     1545 7                                      RETRY_COUNT = .READ_LENGTH;
: 1017     1546 7                                      READ_LENGTH = 1;
: 1018     1547 7                                      END
: 1019     1548 7                                  ELSE
: 1020     1549 7                                      BEGIN
: 1021     1550 6                                          FILE_NUMBER = .VBN - .HEADER_OFFSET[J-1];
: 1022     1551 7                                          IF .IFILEMAP[.FILE_NUMBER-1]
: 1023     1552 7                                              THEN RBLD_MESSAGE (RBLD$_HEADERERR, .FILE_NUMBER, .J, .READ_STATUS);
: 1024     1553 7                                          IF .IFILEMAP[.FILE_NUMBER-1] = 0;
: 1025     1554 7                                              VBN = .VBN + 1;
: 1026     1555 7                                          IF .FILE_NUMBER LEQU FID$C MFD
: 1027     1556 7                                              THEN RBLD_EXIT (RBLD$_SYSHEADER);
: 1028     1557 7                                              ERR_COUNT = .ERR_COUNT + 1;
: 1029     1558 7                                              IF .ERR_COUNT GTRU 10
: 1030     1559 7                                                  THEN RBLD_EXIT (RBLD$_ERRORS);
: 1031     1560 7                                              END;
: 1032     1561 7                                          END;
: 1033     1562 7                                      END;
: 1034     1563 6                                  END;
: 1035     1564 5                              END;
: 1036     1565 4                          END;
: 1037     1566 4                      END;
: 1038     1567 4 ! For each header block that we read, verify that it is a valid file header.
: 1039     1568 4 ! If it is, compute the number of blocks it maps and charge them to the
: 1040     1569 4 ! owner UIC. If the header is not valid, check if it is marked busy in
: 1041     1570 4 ! the index file bitmap. If so, bump the sequence number and write it; then
: 1042     1571 4 ! mark it free. If the write fails, leave it busy.
: 1043     1572 4 !
: 1044     1573 4 !
: 1045     1574 4      BLOCKS_READ = .IO_STATUS[1] / 512;
: 1046     1575 4      HEADER = .BUFFER;
: 1047     1576 4      UNTIL .HEADER GEQA .BUFFER + .BLOCKS_READ * 512
: 1048     1577 4      DO
: 1049     1578 5          BEGIN
: 1050     1579 5              FILE_NUMBER = .VBN + (.HEADER - .BUFFER) / 512 - .HEADER_OFFSET[J-1];
: 1051     1580 5
```

```

: 1052      1581  5      FILE_ID[FID$W_NUM] = .FILE_NUMBER<0,16>;
: 1053      1582  5      FILE_ID[FID$B-NMX] = .FILE_NUMBER<16,8>;
: 1054      1583  5      FILE_ID[FID$W_SEQ] = .HEADER[FH2$W_FID_SEQ];
: 1055      1584  5      STATUS = VERIFY_HEADER (.HEADER, FILE_ID);
: 1056      1585  5      IF .STATUS
: 1057      1586  5      THEN
: 1058      1587  6          BEGIN
: 1059      1588  6              IF FILEMAP[.FILE_NUMBER-1] = 1;
: 1060      1589  6              BLOCK_COUNT = FILE_SIZE (.HEADER) + 1;
: 1061      1590  6              IF .NEED_REBLD [QUOTAS]
: 1062      1591  6              AND .FILE_NUMBER GEQ FID$C_MFD
: 1063      1592  6              THEN COUNT_QUOTA (.HEADER[FH2$L_FILEOWNER], .BLOCK_COUNT, 0);
: 1064      1593  6              END
: 1065      1594  6
: 1066      1595  5      ELSE
: 1067      1596  6          BEGIN
: 1068      1597  6              IF .IFILEMAP[.FILE_NUMBER-1]
: 1069      1598  6              THEN
: 1070      1599  7                  BEGIN
: 1071      1600  7                      IF .STATUS NEQ 2
: 1072      1601  7                      THEN
: 1073      1602  8                          BEGIN
: 1074      1603  8                              CH$FILL (0, 512, .HEADER);
: 1075      1604  8                              $GETTIM (TIMADR = TIME_BUFFER);
: 1076      1605  8                              HEADER[FH2$W_FID_SEQ] = .TIME_BUFFER<16,16>;
: 1077      1606  8                              HEADER[FH2$B_STRUCVER] = 1;
: 1078      1607  8                              HEADER[FH2$B_STRUCLEV] = 2;
: 1079      1608  7                              END;
: 1080      1609  7                              HEADER[FH2$B_IDOFFSET] = FH2$C_LENGTH / 2;
: 1081      1610  7                              HEADER[FH2$B_MPOFFSET] = (FH2$C_LENGTH + FID$C_LENGTH) / 2;
: 1082      1611  7                              HEADER[FH2$B_ACOFFSET] = $BYTEOFFSET (FH2$W_CHECKSUM) / 2;
: 1083      1612  7                              HEADER[FH2$B_RSOFFSET] = $BYTEOFFSET (FH2$W_CHECKSUM) / 2;
: 1084      1613  7                              HEADER[FH2$W_FID_SEQ] = .HEADER[FH2$W_FID_SEQ] + 1;
: 1085      1614  7                              HEADER[FH2$W_FID_NUM] = 0;
: 1086      1615  7                              HEADER[FH2$W_FID_RVN] = 0;
: 1087      1616  7                              HEADER[FH2$W_CHECKSUM] = 0;
: 1088      1617  7                              STATUS = DO_IO (CHAN = .CHANNEL,
: 1089      1618  7                                  FUNC = IOS_WRITEVBLK,
: 1090      1619  7                                  IOSB = IO_STATUS,
: 1091      1620  7                                  P1 = .HEADER,
: 1092      1621  7                                  P2 = 512,
: 1093      1622  7                                  P3 = .FILE_NUMBER + .HEADER_OFFSET[J-1]
: 1094      1623  7                                  );
: 1095      1624  7                              IF .STATUS THEN STATUS = .IO_STATUS[0];
: 1096      1625  7                              IF NOT .STATUS
: 1097      1626  7                              THEN RBLD_MESSAGE (RBLD$WRITEHDR, .FILE_NUMBER, .J, .STATUS)
: 1098      1627  7                              ELSE IFILEMAP[.FILE_NUMBER-1] = 0;
: 1099      1628  6                              END;
: 1100      1629  5                          END;
: 1101      1630  5
: 1102      1631  5          HEADER = .HEADER + 512;
: 1103      1632  4          END;
: 1104      1633  4
: 1105      1634  4      VBN = .VBN + .BLOCKS_READ;
: 1106      1635  4
: 1107      1636  4      IF .READ_LENGTH EQL 1          ! handle single block mode
: 1108      1637  4      THEN
```

```
1109 1638 S BEGIN
1110 1639 S RETRY_COUNT = .RETRY_COUNT - 1;
1111 1640 S IF .RETRY_COUNT EQL 0
1112 1641 S THEN READ_LENGTH = BLOCK_FACTOR;
1113 1642 S END;
1114 1643 S
1115 1644 S END; ! end of processing one volume
1116 1645 S
1117 1646 S Clear unreferenced bits past the index file EOF. Then
1118 1647 S write back new index file bitmap after pass through index file.
1119 1648 S
1120 1649 S INCR I FROM .FILE_NUMBER+1 TO .IFILEMAP_SIZE*8
1121 1650 S DO
1122 1651 S IF IFILEMAP[I-1] = 0;
1123 1652 S
1124 1653 S STATUS = DO_IO (CHAN = .CHANNEL,
1125 1654 S FUNC = IOS_WRITEVBLK,
1126 1655 S IOSB = IO_STATUS,
1127 1656 S P1 = .IFILEMAP,
1128 1657 S P2 = .IFILEMAP_SIZE,
1129 1658 S P3 = .BITMAP_OFFSET[J-1]
1130 1659 S );
1131 1660 S IF .STATUS THEN STATUS = .IO_STATUS[0];
1132 1661 S IF NOT .STATUS
1133 1662 S THEN
1134 1663 S BEGIN
1135 1664 S CLEANUP_FLAGS[CLF_UNLOCK] = 0;
1136 1665 S RBLD_EXIT (RBLD$_RTIBMAP, .J, .STATUS);
1137 1666 S END;
1138 1667 S
1139 1668 S
1140 1669 S Release memory for working copy of indexfile bitmap.
1141 1670 S
1142 1671 S STATUS = LIB$FREE_VM (IFILEMAP_SIZE, IFILEMAP);
1143 1672 S IFILEMAP = 0;
1144 1673 S STATUS = DO_IO (CHAN = .CHANNEL,
1145 1674 S FUNC = IOS_DEACCESS
1146 1675 S );
1147 1676 S
1148 1677 S
1149 1678 S Write out the new storage bitmap then release bitmap buffer.
1150 1679 S
1151 1680 S UPDATE_ALLOCMAP (.J, 1); ! 1 specifies 'erase the data'
1152 1681 S STATUS = LIB$FREE_VM (ALLOCMAP_SIZE, ALLOCMAP); ! release working memory
1153 1682 S ALLOCMAP = 0;
1154 1683 S
1155 1684 S
1156 1685 S Clear the cleanup flag bits in the storage control block.
1157 1686 S
1158 1687 S STATUS = DO_IO (CHAN = .CHANNEL,
1159 1688 S FUNC = IOS_READVBLK,
1160 1689 S IOSB = IO_STATUS,
1161 1690 S P1 = .BOFFER,
1162 1691 S P2 = $12,
1163 1692 S P3 = 1
1164 1693 S );
1165 1694 S IF .STATUS THEN STATUS = .IO_STATUS[0];
```

```
: 1166      1695      3      IF NOT .STATUS
: 1167      1696      3      THEN
: 1168      1697      4      BEGIN
: 1169      1698      4      :      CLEANUP_FLAGS[CLF_UNLOCK] = 0;
: 1170      1699      4      :      RBLD_EXIT (RBLD$_READSCB, .J, .STATUS);
: 1171      1700      4      :      END;
: 1172      1701      3      :
: 1173      1702      3      :      BUFFER[SCBSV_MAPDIRTY2] = 0;
: 1174      1703      3      :      BUFFER[SCBSV_MAPALLOC2] = 0;
: 1175      1704      3      :      BUFFER[SCBSV_FILALLOC2] = 0;
: 1176      1705      3      :      CHECKSUM (.BUFFER);
: 1177      1706      3      :
: 1178      P 1707      3      :      STATUS = DO_IO (CHAN = .CHANNEL,
: 1179      P 1708      3      :      :      FUNC = IOS_WRITEVBLK,
: 1180      P 1709      3      :      :      IOSB = IO_STATUS,
: 1181      P 1710      3      :      :      P1 = .BUFFER,
: 1182      P 1711      3      :      :      P2 = $12,
: 1183      P 1712      3      :      :      P3 = 1
: 1184      1713      3      :      );
: 1185      1714      3      :      IF .STATUS THEN STATUS = .IO_STATUS[0];
: 1186      1715      3      :      IF NOT .STATUS
: 1187      1716      3      :      THEN
: 1188      1717      4      :      BEGIN
: 1189      1718      4      :      :      CLEANUP_FLAGS[CLF_UNLOCK] = 0;
: 1190      1719      4      :      :      RBLD_EXIT (RBLD$_WRITESCIB, .J, .STATUS);
: 1191      1720      4      :      :      END;
: 1192      1721      3      :
: 1193      P 1722      3      :      DO_IO (CHAN = .CHANNEL,
: 1194      P 1723      3      :      :      FUNC = IOS_DEACCESS
: 1195      1724      3      :      :      );
: 1196      1725      3      :
: 1197      1726      3      :
: 1198      1727      3      :      IF .DUALLOC THEN RBLD_MESSAGE (RBLD$_DUALLOC, .J , 0);
: 1199      1728      3      :
: 1200      1729      3      :      Update volume size in VCB now.
: 1201      1730      3      :
: 1202      1731      3      :      STATUS = KERNEL_CALL (SET_FREE, .J);
: 1203      1732      3      :
: 1204      1733      2      :      END;
: 1205      1734      2      :      ! end of processing all volumes
: 1206      1735      2      :
: 1207      1736      2      :      We have now scanned the entire volume set and have a table of the total
: 1208      1737      2      :      disk usage. Take each table entry and use it to update the corresponding
: 1209      1738      2      :      quota file entry. This is done in two passes. The first updates all
: 1210      1739      2      :      existing entries (these come first because of the quota file prescan).
: 1211      1740      2      :      The second pass creates new quota file entries for UIC's that have space
: 1212      1741      2      :      in use but have no quota file entries.
: 1213      1742      2      :
: 1214      1743      2      :      CH$FILL (0, FIB$C_LENGTH, QUOTA_FIB);
: 1215      1744      2      :      QUOTA_FIB[FIB$W_CNTRLFUNC] = FIB$C_MOD_QUOTA;
: 1216      1745      2      :      QUOTA_FIB[FIB$L_CNTRLVAL] = FIB$M_MOD_USE;
: 1217      1746      2      :
: 1218      1747      2      :
: 1219      1748      2      :      Update existing UIC entries in QUOTA.SYS
: 1220      1749      2      :
: 1221      1750      2      :
: 1222      1751      2      :      if .NEED_REBLD [QUOTAS]
```

```
: 1223      1752 2 THEN
: 1224      1753 2 BEGIN
: 1225      1754 2
: 1226      1755 2 local
: 1227      1756 2
: 1228      1757 2 P: ref block[4],
: 1229      1758 2 Q;
: 1230      1759 2
: 1231      1760 2
: 1232      1761 2 Scan entire usage table
: 1233      1762 2
: 1234      1763 2
: 1235      1764 2 Q = .USAGE_TABLE;
: 1236      1765 2
: 1237      1766 2 until .Q geqa .USAGE_TABLE+.TABLE_SIZE do
: 1238      1767 2 begin
: 1239      1768 2
: 1240      1769 2 P = .Q;
: 1241      1770 2
: 1242      1771 2 while .P neq 0 do
: 1243      1772 2 begin
: 1244      1773 2
: 1245      1774 2 if .P[UTB_V_PRESCAN] then
: 1246      1775 2 begin
: 1247      1776 2
: 1248      1777 2 SRC_REC[DQF$L_UIC] = .P[UTB_L_UIC];
: 1249      1778 2 SRC_REC[DQF$L_USAGE] = .P[UTB_L_USAGE];
: 1250      1779 2
: 1251      1780 2 STATUS = DO_IO(chan = .CHANNEL,
: 1252      1781 2 func = IO$ACPCONTROL,
: 1253      1782 2 iosb = IO_STATUS,
: 1254      1783 2 p1 = QFIB_DESC,
: 1255      1784 2 p2 = SRCREC_DESC);
: 1256      1785 2
: 1257      1786 2 if .STATUS then
: 1258      1787 2 STATUS = .IO STATUS[0];
: 1259      1788 2 if not .STATUS then
: 1260      1789 2 RBLD_EXIT(RBLD$MODIFYERR, .STATUS);
: 1261      1790 2
: 1262      1791 2 end;
: 1263      1792 2
: 1264      1793 2 P = .P[UTB_A_NEXT];
: 1265      1794 2
: 1266      1795 2 end;
: 1267      1796 2
: 1268      1797 2 Q = .Q + 16;
: 1269      1798 2
: 1270      1799 2 end;
: 1271      1800 2
: 1272      1801 2 end;
: 1273      1802 2
: 1274      1803 2
: 1275      1804 2 Now we create new quota records for UIC's that are not on the quota file.
: 1276      1805 2 Since the quota file may have to be extended, we must unlock the volume
: 1277      1806 2 at this time. This causes a small timing window in which the UIC's to
: 1278      1807 2 be added could be out of phase, if they resume file activity immediately.
: 1279      1808 2 If you can't tolerate this, just do the rebuild twice.
```

```
1280 1809 2 :
1281 1810 2 :
1282 1811 2 QUOTA_FIB[FIBSW_CNTRLFUNC] = FIBSC_UNLK_VOL;
1283 1812 2 QUOTA_FIB[FIBSL_CNTRLVAL] = 0;
1284 1813 2 STATUS = DO_IO (chan = CHANNEL,
1285 1814 2 FUNC = IOS_ACPCONTROL,
1286 1815 2 IOSB = IO_STATUS,
1287 1816 2 P1 = QFIB_DESC
1288 1817 2 );
1289 1818 2 IF .STATUS THEN STATUS = .IO_STATUS[0];
1290 1819 2 IF NOT .STATUS
1291 1820 2 THEN RBLD_EXIT (RBLDS_UNLOCKERR, .STATUS);
1292 1821 2
1293 1822 2 !CLEANUP_FLAGS[CLF_UNLOCK] = 0;
1294 1823 2
1295 1824 2 QUOTA_FIB[FIBSW_CNTRLFUNC] = FIBSC_ADD_QUOTA;
1296 1825 2
1297 1826 2 :
1298 1827 2 : Enter the new UIC's
1299 1828 2 :
1300 1829 2
1301 1830 2 if .NEED_REBLD [QUOTAS]
1302 1831 2 THEN
1303 1832 2 BEGIN
1304 1833 2 local
1305 1834 2
1306 1835 2 P: ref block[4],
1307 1836 2 Q;
1308 1837 2
1309 1838 2 :
1310 1839 2 Scan entire usage table
1311 1840 2 :
1312 1841 2
1313 1842 2 Q = .USAGE_TABLE;
1314 1843 2
1315 1844 2 until .Q geqa .USAGE_TABLE+.TABLE_SIZE do
1316 1845 2 begin
1317 1846 2
1318 1847 2 P = .Q;
1319 1848 2
1320 1849 2 while .P neq 0 do
1321 1850 2 begin
1322 1851 2
1323 1852 2 if .P[UTB_V_INUSE] and not .P[UTB_V_PRESCAN] then
1324 1853 2 begin
1325 1854 2
1326 1855 2 SRC_REC[DQFSL_UIC] = .P[UTB_L_UIC];
1327 1856 2 SRC_REC[DQFSL_USAGE] = .P[UTB_L_USAGE];
1328 1857 2 SRC_REC[DQFSL_PERMQUOTA] = .DEFAULT_QUOTA;
1329 1858 2 SRC_REC[DQFSL_OVERDRAFT] = .DEFAULT_OVER;
1330 1859 2
1331 1860 2 STATUS = DO_IO(chan = CHANNEL,
1332 1861 2 func = IOS_ACPCONTROL,
1333 1862 2 iosb = IO_STATUS,
1334 1863 2 p1 = QFIB_DESC,
1335 1864 2 p2 = SRCREC_DESC);
1336 1865 2
```

```

        if .STATUS then
            STATUS = .IO STATUS[0];
        if not .STATUS then
            RBLD_EXIT(RBLD$_MODIFYERR, .STATUS);

        end;

        P = .P[UTB_A_NEXT];

    end;

    Q = .Q + 16;

end;

Release the quota table storage.

DELETE_TABLE();

Clear the cleanup flag bits in the storage control block.
We must open the bitmap file on volume 1, read and write the SCB, and
close the file.

QUOTA_FIB[FIB$_ACCTL] = FIB$_WRITE or FIB$_NOWRITE;
QUOTA_FIB[FIB$_FID_RVN] = 1;
QUOTA_FIB[FIB$_FID_NUM] = FID$_BITMAP;
QUOTA_FIB[FIB$_FID_SEQ] = FID$_BITMAP;

STATUS = DO_IO(chan = .CHANNEL,
               func = IOS$_ACCESS or IOSM$_ACCESS,
               iosb = IO STATUS,
               p1 = QFIB$_DESC);

if .STATUS then
    STATUS = .IO STATUS[0];
if not .STATUS then
    RBLD_EXIT(RBLD$_ACCBITMAP, 1, .STATUS);

STATUS = DO_IO(chan = .CHANNEL,
               func = IOS$_READVBLK,
               iosb = IO STATUS,
               p1 = .BUFFER,
               p2 = $12,
               p3 = 1);

if .STATUS then
    STATUS = .IO STATUS[0];
if not .STATUS
    THEN
        RBLD_EXIT(RBLD$_READSCB, 1, .STATUS);

BUFFER[SCB$_V QUODIRTY2] = 0;
CHECKSUM(.BUFFER);

```

```
1394      1923      3
1395      P 1924      3      STATUS = DO_IO(chan = .CHANNEL,
1396      P 1925      3      func = IOS_WRITEVBLK,
1397      P 1926      3      iosb = IO_STATUS,
1398      P 1927      3      p1 = .BUFFER,
1399      P 1928      3      p2 = $12,
1400      1929      3      p3 = 1);
1401      1930      3
1402      1931      3      if .STATUS then
1403      1932      3      STATUS = .IO_STATUS[0];
1404      1933      3      if not .STATUS
1405      1934      3      THEN
1406      1935      3      RBLD_EXIT(RBLD$_WRITESCIB, 1, .STATUS);
1407      1936      3
1408      P 1937      3      STATUS = DO_IO(chan = .CHANNEL,
1409      1938      3      func = IOS_DEACCESS);
1410      1939      3
1411      1940      3      end;
1412      1941      3
1413      1942      3      !
1414      1943      3      ! Free remaining storage and cancel the exit handler.
1415      1944      3      !
1416      1945      3
1417      1946      3      LIB$FREE_VM (UPLIT (BLOCK_FACTOR*512), BUFFER);
1418      1947      3      BUFFER = 0;
1419      1948      3      LIB$FREE_VM (DYN_SIZE, EOF);
1420      1949      3      EOF = 0;
1421      1950      3
1422      1951      3      $CANEXH (DESBK=EXIT_HNDLR_DESC);
1423      1952      3
1424      1953      3      1
1425      1954      3      END;
INFO#250      L1:1538      ! end of routine REBUILD
: Referenced LOCAL symbol FILE_NUMBER is probably not initialized
```

.PSECT \$PLITS,NOWRT,NOEXE,2

```
00008000 00000 P.AAC: .LONG 32768
00008000 00004 P.AAD: .LONG 32768
00008000 00008 P.AAE: .LONG 32768
```

```
.EXTRN CHECKSUM, LIB$FREE VM
.EXTRN LIB$GET VM, SYSS$CANEXH
.EXTRN SYSS$DCLEXH, SYSS$GETTIM
.EXTRN SYSS$CMKRNL
```

.PSECT \$CODE\$,NOWRT,2

```
.ENTRY REBUILD, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,- ; 1011
R11
SUBL2 #60, SP
MOVZWL #1000, DEFAULT_QUOTA ; 1056
MOVZBL #100, DEFAULT_OVER
MOVAL 111$, (FP)
MOVCS #0, (SP), #0, #332, OWN_START ; 1093
```

014C 8F 00

```
5E 3C C2 00002
7E 03E8 8F 3C 00005
7E 64 8F 9A 0000A
6D OFF7 CF DE 0000E
6E 00 2C 00013
```

REBUILD
V04-000

J 7
16-Sep-1984 01:27:55
14-Sep-1984 12:45:34

VAX-11 Bliss-32 V4.0-742
[MOUNT.SRC]REBUILD.B32;2

Page 36
(3)

RE
VO

0040 8F

00

		00000000'	EF	00000000V	EF	9E	0001A		MOVAB	RBLD EXIT_HNDL, EXIT_HNDLR_DESC+4	:	1098
00000000'	EF	00000000'	EF		01	D0	0002A		MOVL	#1, EXIT_HNDLR_DESC+8	:	1099
00000000'	EF	00000000'	EF		9E	00031			MOVAB	EXIT_HNDLR_DESC+16, EXIT_HNDLR_DESC+12	:	1100
		00000000'	EF		9F	0003C			PUSHAB	EXIT_HNDLR_DESC	:	1102
00000000G	00				01	FB	00042		CALLS	#1, SYSSCANEXH	:	
		00000000'	EF		9F	00049			PUSHAB	EXIT_HNDLR_DESC	:	1103
00000000G	00				01	FB	0004F		CALLS	#1, SYSDCCEXH	:	
	5A	40			8F	9A	00056		MOVZBL	#64, READ_LENGTH	:	1109
00000000'	EF	04			AC	B0	0005A		MOVW	CHANNEL_ARG, CHANNEL	:	1115
					0D	12	00062		BNEQ	1\$:	1117
		00450000			8F	DD	00064		PUSHL	#4521984	:	1118
00000000G	00				01	FB	0006A		CALLS	#1, LIB\$STOP	:	
00000000'	EF				07	B0	00071	1\$:	MOVW	#7, QUOTA_FIB+22	:	1123
					7E	7C	00078		CLRQ	-(SP)	:	1128
					7E	7C	0007A		CLRQ	-(SP)	:	
					7E	D4	0007C		CLRL	-(SP)	:	
		00000000'	EF		9F	0007E			PUSHAB	QFIB_DESC	:	
					7E	7C	00084		CLRQ	-(SP)	:	
		00000000'	EF		9F	00086			PUSHAB	IO_STATUS	:	
					38	DD	0008C		PUSHL	#56	:	
	7E	00000000'	EF		3C	0008E			MOVZWL	CHANNEL, -(SP)	:	
					1A	DD	00095		PUSHL	#26	:	
00000000G	00				0C	FB	00097		CALLS	#12, COMMON_IO	:	
	OC				50	D0	0009E		MOVL	R0, STATUS	:	
		OC		OC	AE	E9	000A2		BLBC	STATUS, 2\$:	1129
	OC	AE	00000000'	EF	3C	000A6			MOVZWL	IO_STATUS, STATUS	:	
		10		OC	AE	E8	000AE		BLBS	STATUS, 3\$:	1130
				OC	AE	DD	000B2	2\$:	PUSHL	STATUS	:	1131
		00450020			8F	DD	000B5		PUSHL	#4522016	:	
00000000G	00				02	FB	000BB		CALLS	#2, LIB\$STOP	:	
		00000000'	EF		9F	000C2	3\$:		PUSHAB	BUFFER	:	1138
		00000000'	EF		9F	000C8			PUSHAB	P.AAC	:	
00000000G	00				02	FB	000CE		CALLS	#2, LIB\$GET_VM	:	
	OC				50	D0	000D5		MOVL	R0, STATUS	:	
		OC		OC	AE	E8	000D9		BLBS	STATUS, 4\$:	1139
		00000000'	EF		D4	000DD			CLRL	BUFFER	:	1142
				OC	AE	DD	000E3		PUSHL	STATUS	:	1143
					7E	D4	000E6		CLRL	-(SP)	:	
		00450098			8F	DD	000E8		PUSHL	#4522136	:	
00000000G	00				03	FB	000EE		CALLS	#3, LIB\$STOP	:	
	6E				00	2C	000F5	4\$:	MOVCS	#0, (SP), #0, #64, QUOTA_FIB	:	1149
		00000000'	EF				000FC				:	
00000000'	EF				01	D0	00101		MOVL	#1, QUOTA_FIB	:	1150
00000000'	EF	00010001			8F	D0	00108		MOVL	#65537, QUOTA_FIB+4	:	1151
00000000'	EF				01	B0	00113		MOVW	#1, QUOTA_FIB+8	:	1153
					7E	7C	0011A		CLRQ	-(SP)	:	1159
					7E	7C	0011C		CLRQ			

	OC	OC	OC	AE	E9	00146	BLBC	STATUS, 5\$	1160
		AE	00000000'	EF	3C	0014A	MOVZWL	IO STATUS, STATUS	1161
		12	OC	AE	E8	00152	BLBS	STATUS, 6\$	1162
			OC	AE	DD	00156	PUSHL	STATUS	
				01	DD	00159	PUSHL	#1	
		00450040		8F	DD	0015B	PUSHL	#4522048	
00000000G	00			03	FB	00161	CALLS	#3, LIB\$STOP	
				01	DD	00168	PUSHL	#1	1164
		00000000'		EF	DD	0016A	PUSHL	BUFFER	
00000000V	EF			02	FB	00170	CALLS	#2, READ_HOMEBLOCK	
	50	00000000'		EF	DD	00177	MOVL	BUFFER, R0	1166
	57	28		A0	3C	0017E	MOVZWL	40(R0), VOLUME_COUNT	
				03	12	00182	BNEQ	7\$	1167
	57			01	DD	00184	MOVL	#1, VOLUME_COUNT	
000000FF	8F			57	D1	00187	CML	VOLUME_COUNT, #255	1168
				0D	1B	0018E	BLEQU	8\$	
		00450030		8F	DD	00190	PUSHL	#4522032	1169
00000000G	00			01	FB	00196	CALLS	#1, LIB\$STOP	
	50	07		A7	9E	0019D	MOVAB	7(R7), R0	1174
	50			08	C6	001A1	DIVL2	#8, R0	
00000000' EF	50	00000051		8F	C5	001A4	MULL3	#81, R0, DYN_SIZE	
		00000000'		EF	9F	001B0	PUSHAB	EOF	1175
		00000000'		EF	9F	001B6	PUSHAB	DYN_SIZE	
00000000G	00			02	FB	001BC	CALLS	#2, LIB\$GET_VM	
	OC			50	DD	001C3	MOVL	R0, STATUS	
	18	OC		AE	E8	001C7	BLBS	STATUS, 9\$	1176
		00000000'		EF	D4	001CB	CLRL	EOF	1179
		OC		AE	DD	001D1	PUSHL	STATUS	1180
				7E	D4	001D4	CLRL	-(SP)	
		00450098		8F	DD	001D6	PUSHL	#4522136	
00000000G	00			03	FB	001DC	CALLS	#3, LIB\$STOP	
	56	00000000'		EF	DD	001E3	MOVL	EOF, R6	1183
00000000' EF	6E			00	2C	001EA	MOVCS	#0, (SP), #0, DYN_SIZE, (R6)	
				66		001F3			
00000000'	EF		6647	DE	001F4		MOVAL	(R6)[VOLUME_COUNT], CLUSTER_FACTOR	1184
00000000'	EF	00000000'	FF47	3E	001FC		MOVAV	@CLUSTER_FACTOR[VOLUME_COUNT], -	1185
								HEADER_OFFSET	
00000000'	EF	00000000'	FF47	3E	00208		MOVAV	@HEADER_OFFSET[VOLUME_COUNT], BITMAP_OFFSET	1186
00000000'	EF	00000000'	FF47	3E	00214		MOVAV	@BITMAP_OFFSET[VOLUME_COUNT], -	1187
								VOLUME_PRESENT	
00000000'	FF			01	88	00220	BISB2	#1, @VOLUME_PRESENT	1189
	51	00000000'		EF	DD	00227	MOVL	BUFFER, R1	1190
	50	OE		A1	3C	0022E	MOVZWL	14(R1), R0	
00000000'	FF			50	B0	00232	MOVW	R0, @CLUSTER_FACTOR	
	50			02	78	00239	ASHL	#2, R0, R2	1191
00000000' 52	52			01	A1	0023D	ADDW3	#1, R2, @BITMAP_OFFSET	
00000000' FF	52	20		A1	3C	00245	MOVZWL	32(R1), R2	1192
	53		6240	DE	00249		MOVAL	(R2)[R0], R3	
00000000'	FF			53	B0	0024D	MOVW	R3, @HEADER_OFFSET	
				01	DD	00254	PUSHL	#1	1193
				51	DD	00256	PUSHL	R1	
00000000V	EF			02	FB	00258	CALLS	#2, GET_EOF	
	66			50	DD	0025F	MOVL	R0, (R6)	
				7E	7C	00262	CLRQ	-(SP)	1197
				7E	7C	00264	CLRQ	-(SP)	
				7E	7C	00266	CLRQ	-(SP)	
				7E	7C	00268	CLRQ	-(SP)	

	7E		34	7D	0026A	MOVQ	#52, -(SP)		
	7E	00000000'	EF	3C	0026D	MOVZWL	CHANNEL, -(SP)		
			1A	DD	00274	PUSHL	#26		
10	00000000G	00	0C	FB	00276	CALLS	#12, COMMON_IO		
	08	AC	01	E1	0027D	BBC	#1, BUILD_FLAGS, 10\$	1210	
		7E	01	7D	00282	MOVQ	#1, -(SP)	1212	
	00000000V	EF	02	FB	00285	CALLS	#2, CHECK_SCB_STATUS		
	14	AE	51	DD	0028C	MOVL	R1, NEED_REBLD		
			0B	11	00290	BRB	11\$		
14	AE	14	01	88	00292	BISB2	#1, NEED_REBLD	1215	
01	01	08	AC	F0	00296	INSV	BUILD_FLAGS, #1, #1, NEED_REBLD	1216	
	52		01	DD	0029D	MOVL	#1, J	1224	
			59	11	002A0	BRB	14\$		
	00000000'	EF	52	B0	002A2	MOVW	J, QUOTA_FIB+8	1228	
			7E	7C	002A9	CLRQ	-(SP)	1233	
			7E	7C	002AB	CLRQ	-(SP)		
			7E	D4	002AD	CLRL	-(SP)		
		00000000'	EF	9F	002AF	PUSHAB	QFIB_DESC		
			7E	7C	002B5	CLRQ	-(SP)		
		00000000'	EF	9F	002B7	PUSHAB	IO STATUS		
	7E	72	8F	9A	002BD	MOVZBL	#1T4, -(SP)		
	7E	00000000'	EF	3C	002C1	MOVZWL	CHANNEL, -(SP)		
			1A	DD	002C8	PUSHL	#26		
	00000000G	00	0C	FB	002CA	CALLS	#12, COMMON_IO		
	0C	AE	50	DD	002D1	MOVL	R0, STATUS		
		0C	AE	E9	002D5	BLBC	STATUS, 13\$	1234	
	0C	AE	EF	3C	002D9	MOVZWL	IO STATUS, STATUS		
	19	0C	AE	E8	002E1	BLBS	STATUS, 15\$	1235	
		0C	AE	DD	002E5	PUSHL	STATUS	1238	
			52	DD	002E8	PUSHL	J		
		00450040	8F	DD	002EA	PUSHL	#4522048		
	00000000G	00	03	FB	002F0	CALLS	#3, LIB\$SIGNAL		
	14	AE	02	8A	002F7	BICB2	#2, NEED_REBLD	1239	
			008D	31	002FB	BRW	17\$	1235	
		53	A2	9E	002FE	MOVAB	-1(R2), R3	1243	
00	00000000'	FF	53	E2	00302	BBSS	R3, @VOLUME_PRESENT, 16\$		
			52	DD	0030A	PUSHL	J	1245	
		00000000'	EF	DD	0030C	PUSHL	BUFFER		
	00000000V	EF	02	FB	00312	CALLS	#2, READ_HOMEBLOCK		
	51	00000000'	EF	DD	00319	MOVL	BUFFER, R1	1247	
	50	0E	A1	3C	00320	MOVZWL	14(R1), R0		
	00000000'FF43		50	B0	00324	MOVW	R0, @CLUSTER_FACTOR[R3]		
			02	78	0032C	ASHL	#2, R0, R4	1248	
54			01	A1	00330	ADDW3	#1, R4, @BITMAP_OFFSET[R3]		
00000000'FF43		20	A1	3C	00339	MOVZWL	32(R1), R4	1249	
			6440	DE	0033D	MOVAL	(R4)[R0], R5		
	00000000'FF43		55	B0	00341	MOVW	R5, @HEADER_OFFSET[R3]		
			06	BB	00349	PUSHR	#^M<R1, R2>	1250	
	00000000V	EF	02	FB	0034B	CALLS	#2, GET_EOF		
	00000000'FF43		50	DD	00352	MOVL	R0, @EOF[R3]		
			7E	7C	0035A	CLRQ	-(SP)	1254	
			7E	7C	0035C	CLRQ	-(SP)		
			7E	7C	0035E	CLRQ	-(SP)		
			7E	7C	00360	CLRQ	-(SP)		
	7E		34	7D	00362	MOVQ	#52, -(SP)		
	7E	00000000'	EF	3C	00365	MOVZWL	CHANNEL, -(SP)		
			1A	DD	0036C	PUSHL	#26		

FF11

11	00000000G	00	0C	FB	0036E	CALLS	#12, COMMON IO		
	08	AC	01	E1	00375	BBC	#1, BUILD_FLAGS, 17\$	1259	
		7E	14	AE	9A	MOVZBL	NEED_REBLD, -(SP)	1261	
	00000000V	EF	52	DD	0037E	PUSHL	J		
	14	AE	02	FB	00380	CALLS	#2, CHECK SCB STATUS		
52		01	51	DO	00387	MOVL	R1, NEED_REBLD		
7B	08	AC	57	F1	0038B	17\$:	ACBL	VOLUME COUNT, #1, J	1224
		04	01	E1	00391	BBC	#1, BUILD_FLAGS, 22\$	1272	
	14	AE	08	AC	E8	BLBS	BUILD_FLAGS, 18\$	1276	
	14	AE	02	8A	0039A	BICB2	#2, NEED_REBLD	1278	
04	14	AE	01	E1	0039E	18\$:	BBC	#1, NEED_REBLD, 19\$	1280
	14	AE	01	88	003A3	BISB2	#1, NEED_REBLD	1282	
	00000000'	66	14	AE	E8	19\$:	BLBS	NEED_REBLD, 22\$	1284
		EF	08	B0	003AB	MOVW	#8, QUOTA_FIB+22	1288	
				EF	D4	CLRL	QUOTA_FIB+24	1289	
				7E	7C	CLRQ	-(SP)	1294	
				7E	7C	CLRQ	-(SP)		
				7E	D4	CLRL	-(SP)		
				EF	9F	PUSHAB	QFIB_DESC		
				7E	7C	CLRQ	-(SP)		
				EF	9F	PUSHAB	IO STATUS		
				38	DD	PUSHL	#58		
		7E		EF	3C	MOVZWL	CHANNEL, -(SP)		
				1A	DD	PUSHL	#26		
	00000000G	00	0C	FB	003D7	CALLS	#12, COMMON_IO		
	0C	AE	50	DO	003DE	MOVL	R0, STATUS		
		0C		AE	E9	BLBC	STATUS, 20\$	1296	
	0C	AE	00000000'	EF	3C	MOVZWL	IO STATUS, STATUS		
		10		AE	E8	BLBS	STATUS, 21\$	1298	
				AE	DD	20\$:	PUSHL	STATUS	1299
				8F	DD	PUSHL	#4522024		
	00000000G	00	02	FB	003FB	CALLS	#2, LIB\$STOP		
				EF	9F	21\$:	PUSHAB	BUFFER	1301
				EF	9F	PUSHAB	P.AAD		
				0BC1	31	BRW	110\$		
				EF	D4	CLRL	QUOTA_FIB	1317	
	00000000'	EF	00040004	8F	DO	MOVL	#262148, QUOTA_FIB+10	1318	
	00000000'	EF		01	B0	MOVW	#1, QUOTA_FIB+T4	1320	
	00000000'	EF		09	B0	MOVW	#9, QUOTA_FIB+22	1321	
03	14	AE	01	E0	00430	BBS	#1, NEED_REBLD, 23\$	1322	
			0179	31	00435	BRW	38\$		
				7E	D4	23\$:	CLRL	-(SP)	1331
				EF	9F	PUSHAB	RECATTR_DESC		
				7E	7C	CLRQ	-(SP)		
				EF	9F	PUSHAB	QFILE_NAME		
				EF	9F	PUSHAB	QFIB_DESC		
				7E	7C	CLRQ	-(SP)		
				EF	9F	PUSHAB	IO STATUS		
		7E		8F	9A	MOVZBL	#1T4, -(SP)		
		7E		EF	3C	MOVZWL	CHANNEL, -(SP)		
				1A	DD	PUSHL	#26		
	00000000G	00	0C	FB	00463	CALLS	#12, COMMON_IO		
	0C	AE	50	DO	0046A	MOVL	R0, STATUS		
		0C		AE	E9	BLBC	STATUS, 24\$	1332	
	0C	AE	00000000'	EF	3C	MOVZWL	IO STATUS, STATUS		
		10		AE	E8	BLBS	STATUS, 25\$	1333	
				AE	DD	24\$:	PUSHL	STATUS	1334

50	00000000G	00	00450050	8F	DD	00481	PUSHL	#4522064		
	00000000'	EF		02	FB	00487	CALLS	#2, LIB\$STOP		
	00000000'	EF	FF	10	9C	0048E	25\$:	ROTL	#16, QUOTA_EOF, R0	1336
	00000000V	EF		A0	9E	00496	MOVAB	-1(R0), QUOTA_EOF		
	10	AE		00	FB	0049E	CALLS	#0, ALLOCATE_TABLE		1338
	00000000'	EF	10	01	D0	004A5	MOVL	#1, VBN		1340
				AE	D1	004A9	26\$:	CMPL	VBN, QUOTA_EOF	1341
				03	1B	004B1	BLEQU	27\$		
			00E0	31	004B3	BRW	37\$			
				7E	7C	004B6	27\$:	CLRQ	-(SP)	1350
				7E	D4	004B8	CLRL	-(SP)		
51	00000000'	EF	1C	AE	DD	004BA	PUSHL	VBN		
			20	AE	C3	004BD	SUBL3	VBN, QUOTA_EOF, R1		
				51	D6	004C6	INCL	R1		
		50		5A	D0	004C8	MOVL	READ_LENGTH, R0		
		51		50	D1	004CB	CMPL	R0, R1		
				03	1B	004CE	BLEQU	28\$		
7E		50		51	D0	004D0	MOVL	R1, R0		
		50		09	78	004D3	28\$:	ASHL	#9, R0, -(SP)	
				EF	DD	004D7	PUSHL	BUFFER		
				7E	7C	004DD	CLRQ	-(SP)		
				EF	9F	004DF	PUSHAB	IO STATUS		
				31	DD	004E5	PUSHL	#49		
		7E	00000000'	EF	3C	004E7	MOVZWL	CHANNEL, -(SP)		
				1A	DD	004EE	PUSHL	#26		
	00000000G	00		0C	FB	004F0	CALLS	#12, COMMON_IO		
	OC	AE		50	D0	004F7	MOVL	R0, STATUS		
		OC	OC	AE	E9	004FB	BLBC	STATUS, 29\$		1351
	OC	AE	00000000'	EF	3C	004FF	MOVZWL	IO STATUS, STATUS		
		36	OC	AE	E8	00507	BLBS	STATUS, 32\$		1352
	0000011C	8F	OC	AE	D1	0050B	29\$:	CMPL	STATUS, #284	1355
				1C	12	00513	BNEQ	31\$		
				5A	D7	00515	DECL	READ_LENGTH		1358
				10	12	00517	BNEQ	30\$		1359
			OC	AE	DD	00519	PUSHL	STATUS		1360
			00450058	8F	DD	0051C	PUSHL	#4522072		
00000000G	00			02	FB	00522	CALLS	#2, LIB\$STOP		
			00000000'	EF	B4	00529	30\$:	CLRQ	IO STATUS+2	1361
				10	11	0052F	BRB	32\$		1355
			OC	AE	DD	00531	31\$:	PUSHL	STATUS	1364
			00450058	8F	DD	00534	PUSHL	#4522072		
00000000G	00			02	FB	0053A	CALLS	#2, LIB\$STOP		
	52	00000000'		EF	D0	00541	32\$:	MOVL	BUFFER, ENTRY	1367
	50	00000000'		EF	3C	00548	33\$:	MOVZWL	IO STATUS+2, R0	1368
	50	00000000'		EF	C0	0054F	ADDL2	BUFFER, R0		
				52	D1	00556	CMPL	ENTRY, R0		
				26	1E	00559	BGEQU	36\$		
		1E		62	E9	0055B	BLBC	(ENTRY), 35\$		1372
			04	A2	D5	0055E	TSTL	4(ENTRY)		1375
				0B	12	00561	BNEQ	34\$		
	04	AE	OC	A2	D0	00563	MOVL	12(ENTRY), DEFAULT_QUOTA		1378
		6E	10	A2	D0	00568	MOVL	16(ENTRY), DEFAULT_OVER		1379
				0E	11	0056C	BRB	35\$		1375
				01	DD	0056E	34\$:	PUSHL	#1	1383
				7E	D4	00570	CLRL	-(SP)		
			04	A2	DD	00572	PUSHL	4(ENTRY)		
00000000V	EF			03	FB	00575	CALLS	#3, COUNT_QUOTA		

		52		20	C0	0057C	35\$:	ADDL2	#32, ENTRY	:	1386
				C7	11	0057F		BRB	33\$:	1368
		50	00000000'	EF	3C	00581	36\$:	MOVZWL	IO STATUS+2, R0	:	1388
		50	00000200	8F	C6	00588		DIVL2	#5T2, R0	:	
	10	AE		50	C0	0058F		ADDL2	R0, VEN	:	
				FF	13	00593		BRW	26\$:	1341
				7E	7C	00596	37\$:	CLRQ	-(SP)	:	1393
				7E	7C	00598		CLRQ	-(SP)	:	
				7E	7C	0059A		CLRQ	-(SP)	:	
				7E	7C	0059C		CLRQ	-(SP)	:	
		7E		34	7D	0059E		MOVQ	#52, -(SP)	:	
		7E	00000000'	EF	3C	005A1		MOVZWL	CHANNEL, -(SP)	:	
				1A	DD	005A8		PUSHL	#26	:	
0040	8F	00	00000000G	00	0C	FB 005AA		CALLS	#12, COMMON_IO	:	
		6E		00	2C	005B1	38\$:	MOVCS	#0, (SP), #0, #64, QUOTA_FIB	:	1401
			00000000'	EF		005B8				:	
				58	D4	005BD		CLRL	J	:	1403
				07	1C	005BF	39\$:	BRW	86\$:	
	08	AE		A8	9E	005C2	40\$:	MOVAB	-1(R8), 8(SP)	:	1404
EF	00000000'	FF		AE	E1	005C7		BBC	8(SP), @VOLUME_PRESENT, 39\$:	
			00000000'	EF	D4	005D0		CLRL	DUALLOC	:	1407
				AE	D4	005D6		CLRL	ERR_COUNT	:	1408
	00000000'	EF		01	D0	005D9		MOVL	#1, QUOTA_FIB	:	1409
	00000000'	EF		58	B0	005E0		MOVW	J, QUOTA_FIB+8	:	1410
	00000000'	EF	00020002	8F	D0	005E7		MOVL	#131074, QUOTA_FIB+4	:	1411
				7E	7C	005F2		CLRQ	-(SP)	:	1417
				7E	7C	005F4		CLRQ	-(SP)	:	
				7E	D4	005F6		CLRL	-(SP)	:	
			00000000'	EF	9F	005F8		PUSHAB	QFIB_DESC	:	
				7E	7C	005FE		CLRQ	-(SP)	:	
			00000000'	EF	9F	00600		PUSHAB	IO STATUS	:	
		7E		8F	9A	00606		MOVZBL	#1T4, -(SP)	:	
		7E	00000000'	EF	3C	0060A		MOVZWL	CHANNEL, -(SP)	:	
				1A	DD	00611		PUSHL	#26	:	
00000000G	00			0C	FB	00613		CALLS	#12, COMMON_IO	:	
	0C			50	D0	0061A		MOVL	R0, STATUS	:	
		0C		AE	E9	0061E		BLBC	STATUS, 41\$:	1418
	0C	00000000'		EF	3C	00622		MOVZWL	IO STATUS, STATUS	:	1419
			0C	AE	E8	0062A		BLBS	STATUS, 42\$:	1420
			0C	AE	DD	0062E	41\$:	PUSHL	STATUS	:	
				58	DD	00631		PUSHL	J	:	
			00450048	8F	DD	00633		PUSHL	#4522056	:	
00000000G	00			03	FB	00639		CALLS	#3, LIB\$STOP	:	
				7E	7C	00640	42\$:	CLRQ	-(SP)	:	1428
				01	7D	00642		MOVQ	#1, -(SP)	:	
		7E		8F	3C	00645		MOVZWL	#512, -(SP)	:	
		7E	0200	EF	DD	0064A		PUSHL	BUFFER	:	
			00000000'	7E	7C	00650		CLRQ	-(SP)	:	
				EF	9F	00652		PUSHAB	IO STATUS	:	
			00000000'	31	DD	00658		PUSHL	#49	:	
		7E	00000000'	EF	3C	0065A		MOVZWL	CHANNEL, -(SP)	:	
				1A	DD	00661		PUSHL	#26	:	
00000000G	00			0C	FB	00663		CALLS	#12, COMMON_IO	:	
	0C			50	D0	0066A		MOVL	R0, STATUS	:	
		0C		AE	E9	0066E		BLBC	STATUS, 43\$:	1429
	0C	00000000'		F	3C	00672		MOVZWL	IO STATUS, STATUS	:	
			0C	AE	E8	0067A		BLBS	STATUS, 44\$:	1430

: R

: 1

			0C	AE	DD	0067E	43\$:	PUSHL	STATUS	1434	
				58	DD	00681		PUSHL	J		
			00450068	8F	DD	00683		PUSHL	#4522088		
		00000000G	00	03	FB	00689		CALLS	#3, LIB\$STOP		
			50	08	AE	DD	00690	44\$:	MOVL	8(SP), R0	1437
		00000000'	EF	00000000'	FF	40	3C	00694	MOVZWL	@CLUSTER_FACTOR[R0], ALLOC_CLUSTER	
			50	00000000'	EF	DD	006A0	MOV	BUFFER, R0	1438	
			51	02	A0	3C	006A7	MOVZWL	2(R0), R1		
			51	04	A0	CO	006AB	ADDL2	4(R0), R1		
					51	D7	006AF	DECL	R1		
			52	02	A0	3C	006B1	MOVZWL	2(R0), R2	1439	
			51		52	C6	006B5	DIVL2	R2, R1		
		00000000'	EF	0FFF	C1	9E	006B8	MOVAB	4095(R1), ALLOCMAP_SIZE	1438	
		00000000'	EF	00001000	8F	C7	006C1	DIVL3	#4096, ALLOCMAP_SIZE, R0	1440	
			50		09	78	006CD	ASHL	#9, R0, ALLOCMAP_SIZE		
				00000000'	EF	9F	006D5	PUSHAB	ALLOCMAP	1441	
				00000000'	EF	9F	006DB	PUSHAB	ALLOCMAP_SIZE		
		00000000G	00		02	FB	006E1	CALLS	#2, LIB\$GET_VM		
		OC	AE		50	DD	006E8	MOVL	R0, STATUS		
			16	OC	AE	E8	006EC	BLBS	STATUS, 45\$	1442	
				00000000'	EF	D4	006F0	CLRL	ALLOCMAP	1445	
				OC	AE	DD	006F6	PUSHL	STATUS	1446	
				00450098	8F	DD	006F9	PUSHL	#4522136		
			00		02	FB	006FF	CALLS	#2, LIB\$STOP		
00000000'	EF		00		00	2C	00706	45\$:	MOVCS	#0, (SP), #0, ALLOCMAP_SIZE, @ALLOCMAP	1451
				00000000'	FF		0070F				
			50	00000000'	EF	DD	00714	MOVL	BUFFER, R0	1452	
			51	02	A0	3C	00718	MOVZWL	2(R0), R1		
			51	04	A0	CO	0071F	ADDL2	4(R0), R1		
					51	D7	00723	DECL	R1		
			52	02	A0	3C	00725	MOVZWL	2(R0), R2	1453	
			51		52	C6	00729	DIVL2	R2, R1		
		00000G00'	EF	02	A0	3C	0072C	MOVZWL	2(R0), R3	1454	
			51		53	C5	00730	MULL3	R3, R1, BLOCKS_AVAIL		
			50		01	CE	00738	MNEGL	#1, BITNUMBER	1455	
					08	11	0073B	BRB	47\$		
		00 00000000'	FF		50	E2	0073D	46\$:	BBSS	BITNUMBER, @ALLOCMAP, 47\$	1460
		F4	50		51	F2	00745	47\$:	AOBLSS	R1, BITNUMBER, 46\$	1455
					7E	7C	00749	CLRQ	-(SP)	1465	
					7E	7C	0074B	CLRQ	-(SP)		
					7E	7C	0074D	CLRQ	-(SP)		
					7E	7C	0074F	CLRQ	-(SP)		
			7E		34	7D	00751	MOVQ	#52, -(SP)		
			7E	00000000'	EF	3C	00754	MOVZWL	CHANNEL, -(SP)		
					1A	DD	0075B	PUSHL	#26		
		00000000G	00		0C	FB	0075D	CALLS	#12, COMMON_IO		
		OC	AE		50	DD	00764	MOVL	R0, STATUS		
		00000000'	EF	0101	8F	3C	00768	MOVZWL	#257, QUOTA_FIB	1467	
		00000000'	EF	00010001	8F	DD	00771	MOVL	#65537, QUOTA_FIB+4	1468	
					7E	7C	0077C	CLRQ	-(SP)	1475	
					7E	7C	0077E	CLRQ	-(SP)		
					7E	D4	00780	CLRL	-(SP)		
				00000000'	EF	9F	00782	PUSHAB	QFIB_DESC		
					7E	7C	00788	CLRQ	-(SP)		
				00000000'	EF	9F	0078A	PUSHAB	IO STATUS		
			7E	72	8F	9A	00790	MOVZBL	#1T4, -(SP)		
			7E	00000000'	EF	3C	00794	MOVZWL	CHANNEL, -(SP)		

00000000G	00	1A	DD	0079B	PUSHL	#26			
OC	AE	OC	FB	0079D	CALLS	#12, COMMON_IO			
	OC	50	DO	007A4	MOVL	R0, STATUS			
	AE	AE	E9	007A8	BLBC	STATUS, 48\$	1476		
	OC	EF	3C	007AC	MOVZWL	IO STATUS, STATUS			
	12	AE	E8	007B4	BLBS	STATUS, 49\$	1477		
		OC	DD	007B8	PUSHL	STATUS	1478		
		58	DD	007BB	PUSHL	J			
	00450040	8F	DD	007BD	PUSHL	#4522048			
00000000G	00	03	FB	007C3	CALLS	#3, LIB\$STOP			
	52	08	AE	DO	007CA	49\$: MOVL	8(SP), R2	1482	
	51	00000000'FF	42	3C	007CE	MOVZWL	@HEADER_OFFSET[R2], R1		
	53	08	AE	DO	007D6	MOVL	8(SP), R3		
	50	00000000'FF	43	3C	007DA	MOVZWL	@BITMAP_OFFSET[R3], R0		
50	51		50	C3	007E2	SUBL3	R0, R1, -R0		
50	50		09	78	007E6	ASHL	#9, R0, R0		
00000000'	EF	0200	C0	9E	007EA	MOVAB	512(R0), IFILEMAP_SIZE		
		00000000'	EF	9F	007F3	PUSHAB	IFILEMAP	1483	
		00000000'	EF	9F	007F9	PUSHAB	IFILEMAP_SIZE		
00000000G	00	02	FB	007FF	CALLS	#2, LIB\$GET_VM			
OC	AE	50	DO	00806	MOVL	R0, STATUS			
	16	OC	AE	E8	0080A	BLBS	STATUS, 50\$	1484	
		00000000'	EF	D4	0080E	CLRL	IFILEMAP	1487	
		OC	AE	DD	00814	PUSHL	STATUS	1488	
		00450098	8F	DD	00817	PUSHL	#4522136		
00000000G	00	02	FB	0081D	CALLS	#2, LIB\$STOP			
		7E	7C	00824	50\$: CLRQ	-(SP)	1499		
		7E	D4	00826	CLRL	-(SP)			
	52	14	AE	DO	00828	MOVL	20(SP), R2		
	7E	00000000'FF	42	3C	0082C	MOVZWL	@BITMAP_OFFSET[R2], -(SP)		
	7E	00000000'	EF	7D	00834	MOVQ	IFILEMAP, -(SP)		
			7E	7C	0083B	CLRQ	-(SP)		
		00000000'	EF	9F	0083D	PUSHAB	IO STATUS		
			31	DD	00843	PUSHL	#49		
	7E	00000000'	EF	3C	00845	MOVZWL	CHANNEL, -(SP)		
			1A	DD	0084C	PUSHL	#26		
00000000G	00	OC	FB	0084E	CALLS	#12, COMMON_IO			
OC	AE	50	DO	00855	MOVL	R0, STATUS			
	OC	OC	AE	E9	00859	BLBC	STATUS, 51\$	1500	
	OC	AE	EF	3C	0085D	MOVZWL	IO STATUS, STATUS		
	12	OC	AE	E8	00865	BLBS	STATUS, 52\$	1501	
		OC	AE	DD	00869	PUSHL	STATUS		
			58	DD	0086C	PUSHL	J		
	00450060	8F	DD	0086E	PUSHL	#4522080			
00000000G	00	03	FB	00874	CALLS	#3, LIB\$STOP			
	50	08	AE	DO	0087B	52\$: MOVL	8(SP), R0	1509	
	10	AE	FF	40	3C	0087F	MOVZWL	@HEADER_OFFSET[R0], VBN	
		10	AE	D6	00888	INCL	VBN		
24	AE	08	AE	02	78	0088B	ASHL	#2, 8(SP), 36(SP)	1510
	51	00000000'	EF	AE	C1	00891	ADDL3	36(SP), EOF, R1	
			61	AE	D1	0089A	CMPL	VBN, (R1)	
				03	1B	0089E	BLEQU	54\$	
		0265	31	008A0	BRW	76\$			
			7E	7C	008A3	54\$: CLRQ	-(SP)	1520	
			7E	D4	008A5	CLRL	-(SP)		
		1C	AE	DD	008A7	PUSHL	VBN		
51		61	20	AE	C3	008AA	SUBL3	VBN, (R1), R1	

		51	D6	008AF	INCL	R1	:
	50	5A	D0	008B1	MOVL	READ_LENGTH, R0	:
	51	50	D1	008B4	CMPL	R0, R1	:
		03	1B	008B7	BLEQU	55\$:
	50	51	D0	008B9	MOVL	R1, R0	:
7E	50	09	78	008BC	55\$: ASHL	#9, R0, -(SP)	:
		EF	DD	008C0	PUSHL	BUFFER	:
		7E	7C	008C6	CLRQ	-(SP)	:
		EF	9F	008C8	PUSHAB	IO STATUS	:
		31	DD	008CE	PUSHL	#49	:
	7E	EF	3C	008D0	MOVZWL	CHANNEL, -(SP)	:
		1A	DD	008D7	PUSHL	#26	:
00000000G	00	0C	FB	008D9	CALLS	#12, COMMON IO	:
1C	AE	50	D0	008E0	MOVL	R0, READ_STATUS	:
	0A	AE	E9	008E4	BLBC	READ_STATUS, 56\$	1521
1C	AE	EF	3C	008E8	MOVZWL	IO STATUS, READ_STATUS	:
		06	11	008F0	BRB	57\$:
		EF	D4	008F2	56\$: CLRL	IO STATUS	:
	36	AE	E8	008F8	57\$: BLBS	READ_STATUS, 59\$	1529
		EF	D4	008FC	CLRL	IO STATUS	1532
0000011C	8F	AE	D1	00902	CMPL	READ_STATUS, #284	1533
		1A	12	0090A	BNEQ	58\$:
		5A	D7	0090C	DECL	READ_LENGTH	1536
		22	12	0090E	BNEQ	59\$	1537
		AE	DD	00910	PUSHL	READ_STATUS	1538
	1C	58	DD	00913	PUSHL	J	:
		59	DD	00915	PUSHL	FILE_NUMBER	:
	00450088	8F	DD	00917	PUSHL	#4522120	:
00000000G	00	04	FB	0091D	CALLS	#4, LIB\$STOP	:
		72	11	00924	BRB	64\$	1533
	01	5A	D1	00926	58\$: CMPL	READ_LENGTH, #1	1543
		09	1B	00929	BLEQU	60\$:
28	AE	5A	D0	0092B	MOVL	READ_LENGTH, RETRY_COUNT	1546
	SA	01	D0	0092F	MOVL	#1, READ_LENGTH	1547
		64	11	00932	59\$: BRB	64\$	1543
	50	AE	D0	00934	60\$: MOVL	8(SP), R0	1552
	59	00000000'FF	40	3C	00938	MOVZWL	@HEADER OFFSET[R0], FILE_NUMBER
59	10	AE	59	C3	00940	SUBL3	FILE_NUMBER, VBN, FILE_NUMBER
		52	A9	9E	00945	MOVAB	-1(R9), R2
14	00000000'	FF	52	E1	00949	BBC	R2, @IFILEMAP, 61\$
		1C	AE	DD	00951	PUSHL	READ_STATUS
			58	DD	00954	PUSHL	J
			59	DD	00956	PUSHL	FILE_NUMBER
	00450088	8F	DD	00958	PUSHL	#4522120	:
00000000G	00	04	FB	0095E	CALLS	#4, LIB\$SIGNAL	:
00	00000000'	FF	52	E5	00965	61\$: BBCC	R2, @IFILEMAP, 62\$
		10	AE	D6	0096D	62\$: INCL	VBN
		04	59	D1	00970	CMPL	FILE_NUMBER, #4
			0D	1A	00973	BGTRU	63\$
	004500A8	8F	DD	00975	PUSHL	#4522152	1559
00000000G	00	01	FB	0097B	CALLS	#1, LIB\$STOP	:
		2C	AE	D6	00982	63\$: INCL	ERR_COUNT
	0A	2C	AE	D1	00985	CMPL	ERR_COUNT, #10
			0D	1B	00989	BLEQU	64\$
	004500B0	8F	DD	0098B	PUSHL	#4522160	1562
00000000G	00	01	FB	00991	CALLS	#1, LIB\$STOP	:
	5B	00000000'	EF	3C	00998	64\$: MOVZWL	IO_STATUS+2, BLOCKS_READ

20	AE	5B	00000200	8F	C6	0099F	DIVL2	#512, BLOCKS READ	1575	
		56	00000000	EF	D0	009A6	MOVL	BUFFER, HEADER	1576	
	50	5B		09	78	009AD	ASHL	#9, BLOCKS READ, 32(SP)		
		EF	20	AE	C1	009B2	ADDL3	32(SP), BUFFER, R0		
		50		56	D1	009BB	CMPL	HEADER, R0		
				03	1F	009BE	BLSSU	66\$		
				0130	31	009C0	BRW	74\$		
	50	56	00000000	EF	C3	009C3	SUBL3	BUFFER, HEADER, R0	1580	
		50	00000200	8F	C6	009CB	DIVL2	#512, R0		
		50	10	AE	C0	009D2	ADDL2	VBN, R0		
		51	08	AE	D0	009D6	MOVL	8(SP), R1		
	59	59	00000000	FF41	3C	009DA	MOVZWL	@HEADER OFFSET[R1], FILE NUMBER		
		50		59	C3	009E2	SUBL3	FILE_NUMBER, R0, FILE_NUMBER		
	3C	AE		59	B0	009E6	MOVW	FILE_NUMBER, FILE_ID	1581	
50		08		10	EF	009EA	EXTZV	#16, #8, FILE_NUMBER, R0	1582	
	41	AE		50	90	009EF	MOVB	R0, FILE_ID+5		
	3E	AE		A6	B0	009F3	MOVW	10(HEADER), FILE_ID+2	1583	
			0A	AE	9F	009F8	PUSHAB	FILE_ID	1584	
			3C	56	DD	009FB	PUSHL	HEADER		
	00000000V	EF		02	FB	009FD	CALLS	#2, VERIFY_HEADER		
	0C	AE		50	D0	00A04	MOVL	R0, STATUS		
	18	AE	FF	A9	9E	00A08	MOVAB	-1(R9), 24(SP)	1588	
		33	0C	AE	E9	00A0D	BLBC	STATUS, 69\$	1585	
	00	00000000	FF	18	AE	E2	00A11	BBSS	24(SP), @IFILEMAP, 67\$	1588
				56	DD	00A1A	PUSHL	HEADER	1589	
	00000000V	EF		01	FB	00A1C	CALLS	#1, FILE_SIZE		
	30	AE	01	A0	9E	00A23	MOVAB	1(R0), BLOCK_COUNT		
	14	AE		01	E1	00A28	BBC	#1, NEED REBCD, 68\$	1590	
		04		59	D1	00A2D	CMPL	FILE_NUMBER, #4	1591	
				0F	19	00A30	BLSS	68\$		
				7E	D4	00A32	CLRL	-(SP)	1592	
			34	AE	DD	00A34	PUSHL	BLOCK_COUNT		
			3C	A6	DD	00A37	PUSHL	60(HEADER)		
	00000000V	EF		03	FB	00A3A	CALLS	#3, COUNT_QUOTA		
			00A7	31	00A41	68\$:	BRW	73\$	1585	
	F4	00000000	FF	18	AE	E1	00A44	69\$:	1597	
			02	0C	AE	D1	00A4D	BBC	24(SP), @IFILEMAP, 68\$	1600
				1D	13	00A51	CMPL	STATUS, #2		
0200	8F	00	6E	00	2C	00A53	BEQL	70\$	1603	
				66	00A5A	MOVCS	#0, (SP), #0, #512, (HEADER)			
			34	AE	9F	00A5B	PUSHAB	TIME_BUFFER	1604	
	00000000G	00		01	FB	00A5E	CALLS	#1, SYS\$GETTIM		
	0A	A6	36	AE	B0	00A65	MOVW	TIME_BUFFER+2, 10(HEADER)	1605	
	06	A6	0201	8F	B0	00A6A	MOVW	#513-6(HEADER)	1606	
		66	FFFF6428	8F	D0	00A70	MOVL	#-39896, (HEADER)	1609	
			0A	A6	B6	00A77	INCW	10(HEADER)	1613	
			08	A6	B4	00A7A	CLRW	8(HEADER)	1614	
			0C	A6	B4	00A7D	CLRW	12(HEADER)	1615	
			01FE	C6	B4	00A80	CLRW	510(HEADER)	1616	
				7E	7C	00A84	CLRW	-(SP)	1623	
				7E	D4	00A86	CLRL	-(SP)		
	51	14		AE	D0	00A88	MOVL	20(SP), R1		
	50	00000000	FF41	3C	00A8C	MOVZWL	@HEADER OFFSET[R1], R0			
			6049	9F	00A94	PUSHAB	(R0)[FILE_NUMBER]			
	7E	0200		8F	3C	00A97	MOVZWL	#512, -(SP)		
				56	DD	00A9C	PUSHL	HEADER		
				7E	7C	00A9E	CLRW	-(SP)		

		00000000'	EF	9F	00AA0	PUSHAB	IO STATUS	
			30	DD	00AA6	PUSHL	#48	
	7E	00000000'	EF	3C	00AA8	MOVZWL	CHANNEL, -(SP)	
			1A	DD	00AAF	PUSHL	#26	
00000000G	00		OC	FB	00AB1	CALLS	#12, COMMON_IO	
	OC		50	DD	00AB8	MOVL	R0, STATUS	
		OC	AE	E9	00ABC	BLBC	STATUS, 71\$	1624
	OC	00000000'	EF	3C	00AC0	MOVZWL	IO STATUS, STATUS	
	16		AE	E8	00AC8	BLBS	STATUS, 72\$	1625
		OC	AE	DD	00ACC	PUSHL	STATUS	1626
			58	DD	00ACF	PUSHL	J	
			59	DD	00AD1	PUSHL	FILE_NUMBER	
		00450090	8F	DD	00AD3	PUSHL	#4522128	
00000000G	00		04	FB	00AD9	CALLS	#4, LIB\$SIGNAL	
			09	11	00AE0	BRB	73\$	
00 00000000'	FF	18	AE	E5	00AE2	BBCC	24(SP), @IFILEMAP, 73\$	1627
	56	0200	C6	9E	00AEB	MOVAB	512(R6), HEADER	1631
			FE	B	31	BRW	65\$	1576
	10		5B	C0	00AF3	ADDL2	BLOCKS READ, VBN	1634
			5A	D1	00AF7	CMPL	READ_LENGTH, #1	1636
			09	12	00AFA	BNEQ	75\$	
		28	AE	D7	00AFC	DECL	RETRY_COUNT	1639
			04	12	00AFF	BNEQ	75\$	1640
	5A	40	8F	9A	00B01	MOVZBL	#64, READ_LENGTH	1641
			FD	89	31	BRW	53\$	1510
51 00000000'	EF		03	78	00B08	ASHL	#3, IFILEMAP_SIZE, R1	1649
	50		59	DD	00B10	MOVL	FILE_NUMBER, I	1651
			OC	11	00B13	BRB	78\$	
	52	FF	A0	9E	00B15	MOVAB	-1(R0), R2	
00 00000000'	FF		52	E5	00B19	BBCC	R2, @IFILEMAP, 78\$	
FO	50		51	F3	00B21	AOBLEQ	R1, I, 77\$	
			7E	7C	00B25	CLRQ	-(SP)	1659
			7E	D4	00B27	CLRL	-(SP)	
	52	14	AE	DD	00B29	MOVL	20(SP), R2	
7E 00000000'	FF		42	3C	00B2D	MOVZWL	@BITMAP_OFFSET[R2], -(SP)	
7E 00000000'	EF		7D	00B35	MOVQ	IFILEMAP, -(SP)		
			7E	7C	00B3C	CLRQ	-(SP)	
		00000000'	EF	9F	00B3E	PUSHAB	IO STATUS	
			30	DD	00B44	PUSHL	#48	
	7E	00000000'	EF	3C	00B46	MOVZWL	CHANNEL, -(SP)	
			1A	DD	00B4D	PUSHL	#26	
00000000G	00		OC	FB	00B4F	CALLS	#12, COMMON_IO	
	OC		50	DD	00B56	MOVL	R0, STATUS	
		OC	AE	E9	00B5A	BLBC	STATUS, 79\$	1660
	OC	00000000'	EF	3C	00B5E	MOVZWL	IO STATUS, STATUS	
	12		AE	E8	00B66	BLBS	STATUS, 80\$	1661
		OC	AE	DD	00B6A	PUSHL	STATUS	1665
			58	DD	00B6D	PUSHL	J	
		00450078	8F	DD	00B6F	PUSHL	#4522104	
00000000G	00		03	FB	00B75	CALLS	#3, LIB\$STOP	
		00000000'	EF	9F	00B7C	PUSHAB	IFILEMAP	1671
		00000000'	EF	9F	00B82	PUSHAB	IFILEMAP_SIZE	
00000000G	00		02	FB	00B88	CALLS	#2, LIB\$FREE_VM	
	OC		50	DD	00B8F	MOVL	R0, STATUS	
		00000000'	EF	D4	00B93	CLRL	IFILEMAP	1672
			7E	7C	00B99	CLRQ	-(SP)	1675
			7E	7C	00B9B	CLRQ	-(SP)	

		7E	7C	00B9D	CLRQ	-(SP)	
		7E	7C	00B9F	CLRQ	-(SP)	
	7E		34	7D	00BA1	MOVQ	#52, -(SP)
	7E	00000000'	EF	3C	00BA4	MOVZWL	CHANNEL, -(SP)
			1A	DD	00BAB	PUSHL	#26
00000000G	00		0C	FB	00BAD	CALLS	#12, COMMON_IO
OC	AE		50	DD	00BB4	MOVL	R0, STATUS
			01	DD	00BB8	PUSHL	#1
			58	DD	00BBA	PUSHL	J
00000000V	EF		02	FB	00BBC	CALLS	#2, UPDATE_ALLOCMAP
		00000000'	EF	9F	00BC3	PUSHAB	ALLOCMAP
		00000000'	EF	9F	00BC9	PUSHAB	ALLOCMAP SIZE
00000000G	00		02	FB	00BCF	CALLS	#2, LIB\$FREE_VM
OC	AE		50	DD	00BD6	MOVL	R0, STATUS
		00000000'	EF	D4	00BDA	CLRL	ALLOCMAP
			7E	7C	00BE0	CLRQ	-(SP)
	7E		01	7D	00BE2	MOVQ	#1, -(SP)
	7E	0200	8F	3C	00BE5	MOVZWL	#512, -(SP)
		00000000'	EF	DD	00BEA	PUSHL	BUFFER
			7E	7C	00BF0	CLRQ	-(SP)
		00000000'	EF	9F	00BF2	PUSHAB	IO STATUS
			31	DD	00BF8	PUSHL	#49
	7E	00000000'	EF	3C	00BFA	MOVZWL	CHANNEL, -(SP)
			1A	DD	00C01	PUSHL	#26
00000000G	00		0C	FB	00C03	CALLS	#12, COMMON_IO
OC	AE		50	DD	00C0A	MOVL	R0, STATUS
	OC	OC	AE	E9	00C0E	BLBC	STATUS, 81\$
	OC	00000000'	EF	3C	00C12	MOVZWL	IO STATUS, STATUS
	12	OC	AE	E8	00C1A	BLBS	STATUS, 82\$
		OC	AE	DD	00C1E	PUSHL	STATUS
			58	DD	00C21	PUSHL	J
		00450068	8F	DD	00C23	PUSHL	#4522088
00000000G	00		03	FB	00C29	CALLS	#3, LIB\$STOP
	50	00000000'	EF	DD	00C30	MOVL	BUFFER, R0
1C	A0		07	8A	00C37	BICB2	#7, 28(R0)
			50	DD	00C3B	PUSHL	R0
00000000G	00		01	FB	00C3D	CALLS	#1, CHECKSUM
			7E	7C	00C44	CLRQ	-(SP)
	7E		01	7D	00C46	MOVQ	#1, -(SP)
	7E	0200	8F	3C	00C49	MOVZWL	#512, -(SP)
		00000000'	EF	DD	00C4E	PUSHL	BUFFER
			7E	7C	00C54	CLRQ	-(SP)
		00000000'	EF	9F	00C56	PUSHAB	IO STATUS
			30	DD	00C5C	PUSHL	#48
	7E	00000000'	EF	3C	00C5E	MOVZWL	CHANNEL, -(SP)
			1A	DD	00C65	PUSHL	#26
00000000G	00		0C	FB	00C67	CALLS	#12, COMMON_IO
OC	AE		50	DD	00C6E	MOVL	R0, STATUS
	OC	OC	AE	E9	00C72	BLBC	STATUS, 83\$
	OC	00000000'	EF	3C	00C76	MOVZWL	IO STATUS, STATUS
	12	OC	AE	E8	00C7E	BLBS	STATUS, 84\$
		OC	AE	DD	00C82	PUSHL	STATUS
			58	DD	00C85	PUSHL	J
		00450070	8F	DD	00C87	PUSHL	#4522096
00000000G	00		03	FB	00C8D	CALLS	#3, LIB\$STOP
			7E	7C	00C94	CLRQ	-(SP)
			7E	7C	00C96	CLRQ	-(SP)

1680

1681

1682

1693

1694

1695

1699

1702

1704

1705

1713

1714

1715

1719

1724

0040 F8DE
8F

00000000G	00	00000000'	7E	7C	00C98	CLRQ	-(SP)	1727
	11	00000000'	7E	7C	00C9A	CLRQ	-(SP)	
			34	7D	00C9C	MOVQ	#52, -(SP)	
			EF	3C	00C9F	MOVZWL	CHANNEL, -(SP)	
			1A	DD	00CA6	PUSHL	#26	
			0C	FB	00CA8	CALLS	#12, COMMON_IO	
			EF	E9	00CAF	BLBC	DUALLOC, 85\$	
			7E	D4	00CB6	CLRL	-(SP)	
			58	DD	00CB8	PUSHL	J	
			8F	DD	00CBA	PUSHL	#4522040	
			03	FB	00CC0	CALLS	#3, LIB\$SIGNAL	
			58	DD	00CC7	PUSHL	J	1731
			01	DD	00CC9	PUSHL	#1	
			5E	DD	00CCB	PUSHL	SP	
			EF	9F	00CCD	PUSHAB	SET_FREE	
			04	FB	00CD3	CALLS	#4, -@SYSSCMKRN	
			50	DD	00CDA	MOVL	RO, STATUS	
			57	F1	00CDE	ACBL	VOLUME COUNT, #1, J, 40\$	1404
			00	2C	00CE4	MOVCS	#0, (SP), #0, #64, QUOTA_FIB	1743
			EF		00CEB			
			0D	B0	00CF0	MOVW	#13, QUOTA_FIB+22	1744
			04	DD	00CF7	MOVL	#4, QUOTA_FIB+24	1745
			01	E0	00CFE	BBS	#1, NEED_REBLD, 87\$	1751
			0081	31	00D03	BRW	93\$	
			EF	DD	00D06	MOVL	USAGE_TABLE, Q	1764
			EF	C1	00D0D	ADDL3	TABLE_SIZE, USAGE_TABLE, RO	1766
			53	D1	00D19	CPL	Q, RO	
			69	1E	00D1C	BGEQU	93\$	
			53	DD	00D1E	MOVL	Q, P	1769
			5F	13	00D21	BEQL	92\$	1771
			A2	E9	00D23	BLBC	12(P), 91\$	1774
			62	7D	00D27	MOVQ	(P), SRC_REC+4	1777
			7E	7C	00D2E	CLRQ	-(SP)	1784
			7E	7C	00D30	CLRQ	-(SP)	
			EF	9F	00D32	PUSHAB	SRCREC_DESC	
			EF	9F	00D38	PUSHAB	QFIB_DESC	
			7E	7C	00D3E	CLRL	-(SP)	
			EF	9F	00D40	PUSHAB	IO STATUS	
			38	DD	00D46	PUSHL	#58	
			EF	3C	00D48	MOVZWL	CHANNEL, -(SP)	
			1A	DD	00D4F	PUSHL	#26	
			0C	FB	00D51	CALLS	#12, COMMON_IO	
			50	DD	00D58	MOVL	RO, STATUS	
			AE	E9	00D5C	BLBC	STATUS, 90\$	1786
			EF	3C	00D60	MOVZWL	IO STATUS, STATUS	1787
			AE	E8	00D68	BLBS	STATUS, 91\$	1788
			AE	DD	00D6C	PUSHL	STATUS	1789
			8F	DD	00D6F	PUSHL	#4522000	
			02	FB	00D75	CALLS	#2, LIB\$STOP	
			A2	DD	00D7C	MOVL	8(P), P	1793
			9F	11	00D80	BRB	89\$	1771
			10	CO	00D82	ADDL2	#16, Q	1797
			86	11	00D85	BRB	88\$	1766
			08	B0	00D87	MOVW	#8, QUOTA_FIB+22	1811
			EF	D4	00D8E	CLRL	QUOTA_FIB+24	1812
			7E	7C	00D94	CLRL	-(SP)	1817
			7E	7C	00D96	CLRL	-(SP)	

		7E	D4	00D98	CLRL	-(SP)	
	00000000'	EF	9F	00D9A	PUSHAB	QFIB_DESC	
		7E	7C	00DA0	CLRQ	-(SP)	
	00000000'	EF	9F	00DA2	PUSHAB	IO STATUS	
		38	DD	00DA8	PUSHL	#58	
	7E 00000000'	EF	3C	00DAA	MOVZWL	CHANNEL, -(SP)	
		1A	DD	00DB1	PUSHL	#26	
00000000G	00	0C	FB	00DB3	CALLS	#12, COMMON_IO	
OC	AE	50	D0	00DBA	MOVL	RO, STATUS	
	OC	AE	E9	00DBE	BLBC	STATUS, 94\$	1818
	OC	AE	3C	00DC2	MOVZWL	IO STATUS, STATUS	
	10	AE	E8	00DCA	BLBS	STATUS, 95\$	1819
		AE	DD	00DCE	PUSHL	STATUS	1820
	00450028	8F	DD	00DD1	PUSHL	#4522024	
00000000G	00	02	FB	00DD7	CALLS	#2, LIB\$STOP	
00000000'	EF	0B	B0	00DDE	MOVW	#11, QUOTA_FIB+22	1824
03	14	AE	01	00DE5	BBS	#1, NEED_REBLD, 96\$	1830
		01D9	31	00DEA	BRW	109\$	
	53 00000000'	EF	D0	00DED	MOVL	USAGE_TABLE, Q	1842
50	00000000'	EF	C1	00DF4	ADDL3	TABLE_SIZE, USAGE_TABLE, RO	1844
	50	53	D1	00E00	CMPL	Q, RO	
		7E	1E	00E03	BGEQU	102\$	
	52	53	D0	00E05	MOVL	Q, P	1847
		73	13	00E08	BEQL	101\$	1849
68	OC	01	E1	00E0A	BBC	#1, 12(P), 100\$	1852
		A2	E8	00E0F	BLBS	12(P), 100\$	
	00000000'	EF	62	7D	MOVQ	(P), SRC_REC+4	1855
	00000000'	EF	AE	D0	MOVL	DEFAULT_QUOTA, SRC_REC+12	1857
	00000000'	EF	6E	D0	MOVL	DEFAULT_OVER, SRC_REC+16	1858
		7E	7C	00E29	CLRQ	-(SP)	1864
		7E	7C	00E2B	CLRQ	-(SP)	
	00000000'	EF	9F	00E2D	PUSHAB	SRCREC_DESC	
	00000000'	EF	9F	00E33	PUSHAB	QFIB_DESC	
		7E	7C	00E39	CLRQ	-(SP)	
	00000000'	EF	9F	00E3B	PUSHAB	IO STATUS	
		38	DD	00E41	PUSHL	#58	
	7E 00000000'	EF	3C	00E43	MOVZWL	CHANNEL, -(SP)	
		1A	DD	00E4A	PUSHL	#26	
00000000G	00	0C	FB	00E4C	CALLS	#12, COMMON_IO	
OC	AE	50	D0	00E53	MOVL	RO, STATUS	
	OC	AE	E9	00E57	BLBC	STATUS, 99\$	1866
	OC	AE	3C	00E5B	MOVZWL	IO STATUS, STATUS	1867
	10	AE	E8	00E63	BLBS	STATUS, 100\$	1868
		AE	DD	00E67	PUSHL	STATUS	1869
	00450010	8F	DD	00E6A	PUSHL	#4522000	
00000000G	00	02	FB	00E70	CALLS	#2, LIB\$STOP	
	52	08	A2	D0	MOVL	8(P), P	1873
		8B	11	00E7B	BRB	98\$	1849
	53	10	C0	00E7D	ADDL2	#16, Q	1877
		FF71	31	00E80	BRW	97\$	1844
00000000V	EF	00	FB	00E83	CALLS	#0, DELETE_TABLE	1885
00000000'	EF	0101	8F	3C	MOVZWL	#257, QUOTA_FIB	1893
00000000'	EF	02	B0	00E93	MOVW	#2, QUOTA_FIB+4	1895
00000000'	EF	00010002	8F	D0	MOVL	#65538, QUOTA_FIB+6	1896
		7E	7C	00EA5	CLRQ	-(SP)	1901
		7E	7C	00EA7	CLRQ	-(SP)	
		7E	D4	00EA9	CLRL	-(SP)	

		00000000'	EF 9F 00EAB	PUSHAB	QFIB_DESC	
			7E 7C 00EB1	CLRQ	-(SP)	
		00000000'	EF 9F 00EB3	PUSHAB	IO_STATUS	
	7E	72	8F 9A 00EB9	MOVZBL	#1T4, -(SP)	
	7E	00000000'	EF 3C 00EBD	MOVZWL	CHANNEL, -(SP)	
			1A DD 00EC4	PUSHL	#26	
00000000G	00		OC FB 00EC6	CALLS	#12, COMMON_IO	
OC	AE		50 DO 00ECD	MOVL	R0, STATUS	
	OC	OC	AE E9 00ED1	BLBC	STATUS, 103\$	1903
OC	AE	00000000'	EF 3C 00ED5	MOVZWL	IO_STATUS, STATUS	1904
	12	OC	AE E8 00EDD	BLBS	STATUS, 104\$	1905
		OC	AE DD 00EE1	PUSHL	STATUS	1906
			01 DD 00EE4	PUSHL	#1	
		00450048	8F DD 00EE6	PUSHL	#4522056	
00000000G	00		03 FB 00EEC	CALLS	#3, LIB\$STOP	
			7E 7C 00EF3	CLRQ	-(SP)	1913
	7E		01 7D 00EF5	MOVQ	#1, -(SP)	
	7E	0200	8F 3C 00EF8	MOVZWL	#512, -(SP)	
		00000000'	EF DD 00EFD	PUSHL	BUFFER	
			7E 7C 00F03	CLRQ	-(SP)	
		00000000'	EF 9F 00F05	PUSHAB	IO_STATUS	
			31 DD 00F0B	PUSHL	#49	
	7E	00000000'	EF 3C 00F0D	MOVZWL	CHANNEL, -(SP)	
			1A DD 00F14	PUSHL	#26	
00000000G	00		OC FB 00F16	CALLS	#12, COMMON_IO	
OC	AE		50 DO 00F1D	MOVL	R0, STATUS	
	OC	OC	AE E9 00F21	BLBC	STATUS, 105\$	1915
OC	AE	00000000'	EF 3C 00F25	MOVZWL	IO_STATUS, STATUS	1916
	12	OC	AE E8 00F2D	BLBS	STATUS, 106\$	1917
		OC	AE DD 00F31	PUSHL	STATUS	1919
			01 DD 00F34	PUSHL	#1	
		00450068	8F DD 00F36	PUSHL	#4522088	
00000000G	00		03 FB 00F3C	CALLS	#3, LIB\$STOP	
	50	00000000'	EF DD 00F43	MOVL	BUFFER, R0	1921
1C	A0		08 8A 00F4A	BICB2	#8, 28(R0)	
			50 DD 00F4E	PUSHL	R0	1922
00000000G	00		01 FB 00F50	CALLS	#1, CHECKSUM	
			7E 7C 00F57	CLRQ	-(SP)	1929
	7E		01 7D 00F59	MOVQ	#1, -(SP)	
	7E	0200	8F 3C 00F5C	MOVZWL	#512, -(SP)	
		00000000'	EF DD 00F61	PUSHL	BUFFER	
			7E 7C 00F67	CLRQ	-(SP)	
		00000000'	EF 9F 00F69	PUSHAB	IO_STATUS	
			30 DD 00F6F	PUSHL	#48	
	7E	00000000'	EF 3C 00F71	MOVZWL	CHANNEL, -(SP)	
			1A DD 00F78	PUSHL	#26	
00000000G	00		OC FB 00F7A	CALLS	#12, COMMON_IO	
OC	AE		50 DO 00F81	MOVL	R0, STATUS	
	OC	OC	AE E9 00F85	BLBC	STATUS, 107\$	1931
OC	AE	00000000'	EF 3C 00F89	MOVZWL	IO_STATUS, STATUS	1932
	12	OC	AE E8 00F91	BLBS	STATUS, 108\$	1933
		OC	AE DD 00F95	PUSHL	STATUS	1935
			01 DD 00F98	PUSHL	#1	
		00450070	8F DD 00F9A	PUSHL	#4522096	
00000000G	00		03 FB 00FA0	CALLS	#3, LIB\$STOP	
			7E 7C 00FA7	CLRQ	-(SP)	1938
			7E 7C 00FA9	CLRQ	-(SP)	

		7E	7C	00FAB	CLRQ	-(SP)	
		7E	7C	00FAD	CLRQ	-(SP)	
	7E		34	7D 00FAF	MOVQ	#52, -(SP)	
	7E	00000000'	EF	3C 00FB2	MOVZWL	CHANNEL, -(SP)	
			1A	DD 00FB9	PUSHL	#26	
00000000G	00		0C	FB 00FBB	CALLS	#12, COMMON_IO	
	0C	AE	50	D0 00FC2	MOVL	R0, STATUS	
		00000000'	EF	9F 00FC6	PUSHAB	BUFFER	1946
		00000000'	EF	9F 00FCC	PUSHAB	P.AAE	
00000000G	00		02	FB 00FD2	CALLS	#2, LIB\$FREE_VM	
		00000000'	EF	D4 00FD9	CLRL	BUFFER	1947
		00000000'	EF	9F 00FDF	PUSHAB	EOF	1948
		00000000'	EF	9F 00FE5	PUSHAB	DYN_SIZE	
00000000G	00		02	FB 00FEB	CALLS	#2, LIB\$FREE_VM	
		00000000'	EF	D4 00FF2	CLRL	EOF	1949
		00000000'	EF	9F 00FF8	PUSHAB	EXIT_HNDLR_DESC	1951
00000000G	00		01	FB 00FFE	CALLS	#1, SYSSCANEXH	
	50		01	D0 01005	MOVL	#1, R0	1954
				04 01008	RET		
				0000 01009	.WORD	Save nothing	1056
			7E	D4 0100B	CLRL	-(SP)	
			5E	DD 0100D	PUSHL	SP	
	7E	04	AC	7D 0100F	MOVQ	4(AP), -(SP)	
00000000V	EF		03	FB 01013	CALLS	#3, RBLD_HANDLER	
			04	0101A	RET		

; Routine Size: 4123 bytes, Routine Base: \$CODE\$ + 004B

```
1427 1955 1 ROUTINE CHECK_SCB_STATUS (RVN, FLAGS_IN ; FLAGS) : NOVALUE L_ONE_ARG_OUT =
1428 1956 1
1429 1957 1 ++
1430 1958 1
1431 1959 1 Functional Description:
1432 1960 1
1433 1961 1 This routine opens the storage bitmap, reads the storage
1434 1962 1 control block, and then OR's appropriate rebuild flags.
1435 1963 1
1436 1964 1 Makes use of/alters:
1437 1965 1
1438 1966 1 QUOTA FIB
1439 1967 1 CHANNEL
1440 1968 1 BUFFER
1441 1969 1 IO_STATUS
1442 1970 1
1443 1971 1 --
1444 1972 1
1445 1973 2 BEGIN
1446 1974 2
1447 1975 2 MAP
1448 1976 2 FLAGS : BITVECTOR [2];
1449 1977 2
1450 1978 2 LOCAL
1451 1979 2 SBM_FIB : BBLOCK [FIB$C_LENGTH],
1452 1980 2 SBM_FIB_D : VECTOR [2] INITIAL (FIB$C_LENGTH, SBM_FIB),
1453 1981 2 STATUS;
1454 1982 2
1455 1983 2 FLAGS = .FLAGS_IN<0,2,0>;
1456 1984 2
1457 1985 2 CH$FILL (0, FIB$C_LENGTH, SBM_FIB);
1458 1986 2
1459 1987 2 SBM_FIB[FIB$L_ACCTL] = FIB$M_NOWRITE;
1460 1988 2 SBM_FIB[FIB$W_FID_RVN] = .RVN;
1461 1989 2 SBM_FIB[FIB$W_FID_NUM] = FIB$C_BITMAP;
1462 1990 2 SBM_FIB[FIB$W_FID_SEQ] = FIB$C_BITMAP;
1463 P 1991 2 STATUS = DO_IO (CHAN = .CHANNEL,
1464 P 1992 2 FUNC = IO$ACCESS OR IO$M_ACCESS,
1465 P 1993 2 IOSB = IO_STATUS,
1466 P 1994 2 P1 = SBM_FIB_D
1467 1995 2 );
1468 1996 2
1469 1997 2 IF .STATUS THEN STATUS = .IO_STATUS[0];
1470 1998 2
1471 1999 2 IF NOT .STATUS
1472 2000 2 THEN RBLD_EXIT (RBLD$ACCBITMAP, .RVN, .STATUS);
1473 2001 2
1474 P 2002 2 STATUS = DO_IO (CHAN = .CHANNEL,
1475 P 2003 2 FUNC = IO$READVBLK,
1476 P 2004 2 IOSB = IO_STATUS,
1477 P 2005 2 P1 = .BUFFER,
1478 P 2006 2 P2 = 512,
1479 P 2007 2 P3 = 1
1480 2008 2 );
1481 2009 2
1482 2010 2 IF .STATUS THEN STATUS = .IO_STATUS[0];
1483 2011 2
```

REB
V04

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	-----

1955
1973
1983
1985
1987
1988
1989
1995
1997

0E		52	E8	00067	BLBS	STATUS, 2\$	1999
		52	DD	0006A	1\$: PUSHL	STATUS	2000
	04	AC	DD	0006C	PUSHL	RVN	
69	00450048	8F	DD	0006F	PUSHL	#4522056	
		03	FB	00075	CALLS	#3, LIB\$STOP	
		7E	7C	00078	2\$: CLRQ	-(SP)	2008
7E		01	7D	0007A	MOVQ	#1, -(SP)	
7E	0200	8F	3C	0007D	MOVZWL	#512, -(SP)	
	D8	A7	DD	00082	PUSHL	BUFFER	
		7E	7C	00085	CLRQ	-(SP)	
		57	DD	00087	PUSHL	R7	
		31	DD	00089	PUSHL	#49	
7E	FA	A7	3C	0008B	MOVZWL	CHANNEL, -(SP)	
		1A	DD	0008F	PUSHL	#26	
68		0C	FB	00091	CALLS	#12, COMMON_IO	
52		50	D0	00094	MOVL	R0, STATUS	
06		52	E9	00097	BLBC	STATUS, 3\$	2010
52		67	3C	0009A	MOVZWL	IO STATUS, STATUS	
0E		52	E8	0009D	BLBS	STATUS, 4\$	2012
		52	DD	000A0	3\$: PUSHL	STATUS	2014
	04	AC	DD	000A2	PUSHL	RVN	
	00450068	8F	DD	000A5	PUSHL	#4522088	
69		03	FB	000AB	CALLS	#3, LIB\$STOP	
50	D8	A7	D0	000AE	4\$: MOVL	BUFFER, R0	2020
05	1C	A0	01	E0	000B2	BBS	#1, 28(R0), 5\$
03	1C	A0	02	E1	000B7	BBC	#2, 28(R0), 6\$
		56	01	88	000BC	5\$: BISB2	#1, FLAGS
09	1C	A0	03	E1	000BF	6\$: BBC	#3, 28(R0), 7\$
		01	04	AC	D1	000C4	CMPL
		03	12	000C8	BNEQ	7\$	
56		02	88	000CA	BISB2	#2, FLAGS	2031
		7E	7C	000CD	7\$: CLRQ	-(SP)	2034
		7E	7C	000CF	CLRQ	-(SP)	
		7E	7C	000D1	CLRQ	-(SP)	
		7E	7C	000D3	CLRQ	-(SP)	
7E		34	7D	000D5	MOVQ	#52, -(SP)	
7E	FA	A7	3C	000D8	MOVZWL	CHANNEL, -(SP)	
		1A	DD	000DC	PUSHL	#26	
68		0C	FB	000DE	CALLS	#12, COMMON_IO	
51		56	D0	000E1	MOVL	R6, R1	2036
		04	000E4	RET			

; Routine Size: 229 bytes, Routine Base: \$CODE\$ + 1066

; 1509 2037 1

```
: 1511 2038 1 ROUTINE GET_EOF (BUFFER, RVN) =
: 1512 2039 1
: 1513 2040 1 ++
: 1514 2041 1
: 1515 2042 1 Functional Description:
: 1516 2043 1
: 1517 2044 1 This routine finds out the actual EOF of the currently open index
: 1518 2045 1 file by (1) reading the index file record attributes and (2) looking
: 1519 2046 1 for the highest active file header as per the index file bitmap
: 1520 2047 1 and taking the maximum. This is minimized against the actual file
: 1521 2048 1 size.
: 1522 2049 1
: 1523 2050 1 Calling Sequence:
: 1524 2051 1 standard
: 1525 2052 1
: 1526 2053 1 Input Parameters:
: 1527 2054 1 none
: 1528 2055 1
: 1529 2056 1 Implicit Inputs:
: 1530 2057 1 none
: 1531 2058 1
: 1532 2059 1 Output Parameters:
: 1533 2060 1 none
: 1534 2061 1
: 1535 2062 1 Implicit Outputs:
: 1536 2063 1 none
: 1537 2064 1
: 1538 2065 1 Routines Called:
: 1539 2066 1 none
: 1540 2067 1
: 1541 2068 1 Routine Value:
: 1542 2069 1 none
: 1543 2070 1
: 1544 2071 1 Signals:
: 1545 2072 1 none
: 1546 2073 1
: 1547 2074 1 Side Effects:
: 1548 2075 1 none
: 1549 2076 1
: 1550 2077 1 --
: 1551 2078 1
: 1552 2079 2 BEGIN
: 1553 2080 2
: 1554 2081 2 BUILTIN
: 1555 2082 2 ROT;
: 1556 2083 2
: 1557 2084 2 MAP
: 1558 2085 2 BUFFER : REF BBLOCK; ! block buffer arg
: 1559 2086 2
: 1560 2087 2 LOCAL
: 1561 2088 2 STATUS, ! general status value
: 1562 2089 2 BITMAP_SIZE, ! number of blocks in index file bitmap
: 1563 2090 2 HIBLK, ! saved copy of index file HIBLK
: 1564 2091 2 EOF; ! computed index EOF
: 1565 2092 2
: 1566 2093 2 EXTERNAL ROUTINE
: 1567 2094 2 LEFT_ONE; ! find leftmost one bit in a word
```

```
1568 2095 2
1569 2096
1570 2097 ! Read the index file header and the the EOF stored therein.
1571 2098
1572 2099
1573 2100 BITMAP_SIZE = .BUFFER[HM2$W_IBMAPSIZE];
1574 2101 STATUS = DO_IO (CHAN = .CHANNEL,
1575 2102             FUNC = IOS_READVBLK,
1576 2103             IOSB = IO_STATUS,
1577 2104             P1 = .BOFFER,
1578 2105             P2 = 512,
1579 2106             P3 = .HEADER_OFFSET[.RVN-1] + 1
1580 2107             );
1581 2108 IF .STATUS THEN STATUS = .IO_STATUS[0];
1582 2109 IF NOT .STATUS
1583 2110 THEN RBLD_EXIT (RBLD$_BITMAPERR, .RVN, .STATUS);
1584 2111
1585 2112 EOF = ROT (.BBLOCK [BUFFER[FH2$W_RECATTR], FAT$L_EFBLK], 16) - 1;
1586 2113 HIBLK = ROT (.BBLOCK [BUFFER[FH2$W_RECATTR], FAT$L_HIBLK], 16);
1587 2114
1588 2115 ! Now scan the volume's index file bitmap backwards, looking for the highest
1589 2116 ! bit set.
1590 2117
1591 2118
1592 2119 DECR J FROM .BITMAP_SIZE - 1 TO 0
1593 2120 DO
1594 2121 BEGIN
1595 2122 MAP BUFFER : REF VECTOR;
1596 2123 STATUS = DO_IO (CHAN = .CHANNEL,
1597 2124             FUNC = IOS_READVBLK,
1598 2125             IOSB = IO_STATUS,
1599 2126             P1 = .BOFFER,
1600 2127             P2 = 512,
1601 2128             P3 = .BITMAP_OFFSET[.RVN-1] + .J
1602 2129             );
1603 2130 IF .STATUS THEN STATUS = .IO_STATUS[0];
1604 2131 IF NOT .STATUS
1605 2132 THEN RBLD_EXIT (RBLD$_BITMAPERR, .RVN, .STATUS);
1606 2133
1607 2134 DECR I FROM 127 TO 0
1608 2135 DO
1609 2136 BEGIN
1610 2137 IF .BUFFER[.I] NEQ 0
1611 2138 THEN
1612 2139 BEGIN
1613 2140 EOF = MAXU (.J*4096 + .I*32 + LEFT_ONE (.BUFFER[.I])
1614 2141             + .HEADER_OFFSET[.RVN-1],
1615 2142             .EOF);
1616 2143 RETURN MINU (.EOF, .HIBLK);
1617 2144 END;
1618 2145 END;
1619 2146 END;
1620 2147
1621 2148 0
1622 2149 1 END; ! end of routine GET_EOF
```

		.EXTRN		LEFT_ONE		
		OFFC	00000	GET_EOF:	WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11
58	00000000G	00	9E 00002	MOVAB	COMMON_IO, R11	2038
5A	00000000	EF	9E 00009	MOVAB	IO STATUS, R10	
57	04	AC	DD 00010	MOVL	BUFFER, R7	2100
53	20	A7	3C 00014	MOVZWL	32(R7), BITMAP_SIZE	
		7E	7C 00018	CLRQ	-(SP)	2107
		7E	D4 0001A	CLRL	-(SP)	
56	08	AC	DD 0C01C	MOVL	RVN, R6	
50	0118	DA	46 3E 00020	MOVAB	@HEADER_OFFSET[R6], R0	
7E	FE	A0	3C 00026	MOVZWL	-2(R0), -(SP)	
		6E	D6 0002A	INCL	(SP)	
7E	0200	8F	3C 0002C	MOVZWL	#512, -(SP)	
		57	DD 00031	PUSHL	R7	
		7E	7C 00033	CLRQ	-(SP)	
		5A	DD 00035	PUSHL	R10	
		31	DD 00037	PUSHL	#49	
7E	FA	AA	3C 00039	MOVZWL	CHANNEL, -(SP)	
		1A	DD 0003D	PUSHL	#26	
6B		0C	FB 0003F	CALLS	#12, COMMON_IO	
58		50	DD 00042	MOVL	R0, STATUS	
06		58	E9 00045	BLBC	STATUS, 1\$	2108
58		6A	3C 00048	MOVZWL	IO STATUS, STATUS	
12		58	E8 0004B	BLBS	STATUS, 2\$	2109
		58	DD 0004E	PUSHL	STATUS	2110
	08	AC	DD 00050	PUSHL	RVN	
	00450060	8F	DD 00053	PUSHL	#4522080	
		03	FB 00059	CALLS	#3, LIB\$STOP	
50	00000000G	00				
	1C	A7	10 9C 00060	2\$: ROTL	#16, 28(R7), R0	2112
		54	FF A0 9E 00065	MOVAB	-1(R0), EOF	
59	18	A7	10 9C 00069	ROTL	#16, 24(R7), HIBLK	2113
		008E	31 0006E	BRW	9\$	2119
		7E	7C 00071	3\$: CLRQ	-(SP)	2129
		7E	D4 00073	CLRL	-(SP)	
50	011C	DA	46 3E 00075	MOVAB	@BITMAP_OFFSET[R6], R0	
50		70	3C 0007B	MOVZWL	-(R0), R0	
		6340	9F 0007E	PUSHAB	(J)[R0]	
7E	0200	8F	3C 00081	MOVZWL	#512, -(SP)	
		57	DD 00086	PUSHL	R7	
		7E	7C 00088	CLRQ	-(SP)	
		5A	DD 0008A	PUSHL	R10	
		31	DD 0008C	PUSHL	#49	
7E	FA	AA	3C 0008E	MOVZWL	CHANNEL, -(SP)	
		1A	DD 00092	PUSHL	#26	
6B		0C	FB 00094	CALLS	#12, COMMON_IO	
58		50	DD 00097	MOVL	R0, STATUS	
06		58	E9 0009A	BLBC	STATUS, 4\$	2130
58		6A	3C 0009D	MOVZWL	IO STATUS, STATUS	
12		58	E8 000A0	BLBS	STATUS, 5\$	2131
		58	DD 000A3	4\$: PUSHL	STATUS	2132
	08	AC	DD 000A5	PUSHL	RVN	
	00450060	8F	DD 000A8	PUSHL	#4522080	
		03	FB 000AE	CALLS	#3, LIB\$STOP	
00000000G	00					
	52	7F	8F 9A 000B5	5\$: MOVZBL	#127, I	2134
		6742	D5 000B9	6\$: TSTL	(R7)[1]	2137

50	53	3E 13 000BC	BEQL	8\$:	2140
51	52	0C 78 000BE	ASHL	#12, J, R0	:	
55	50	05 78 000C2	ASHL	#5, I, R1	:	
		51 C1 000C6	ADDL3	R1, R0, R5	:	
		6742 DD 000CA	PUSHL	(R7)[I]	:	
51 00000000G	00	01 FB 000CD	CALLS	#1, LEFT ONE	:	
	55	50 C1 000D4	ADDL3	R0, R5, R1	:	
	50	0118 DA46 3E 000D8	MOVAV	@HEADER_OFFSET[R6], R0	:	2141
	55	FE A0 3C 000DE	MOVZWL	-2(R0), R5	:	
	51	55 C0 000E2	ADDL2	R5, R1	:	
	54	51 D1 000E5	CMPL	R1, EOF	:	2142
		03 1E 000E8	BGEQU	7\$:	
	51	54 D0 000EA	MOVL	EOF, R1	:	
	54	51 D0 000ED 7\$:	MOVL	R1, EOF	:	2140
	50	54 D0 000F0	MOVL	EOF, R0	:	2143
	59	50 D1 000F3	CMPL	R0, HIBLK	:	
		11 1B 000F6	BLEQU	12\$:	
	50	59 D0 000F8	MOVL	HIBLK, R0	:	
		04 000FB	RET		:	
	BA	52 F4 000FC 8\$:	SOBGEQ	I, 6\$:	2134
	02	53 F4 000FF 9\$:	SOBGEQ	J, 10\$:	2119
		03 11 00102	BRB	11\$:	
		FF6A 31 00104 10\$:	BRW	3\$:	
		50 D4 00107 11\$:	CLRL	R0	:	2149
		04 00109 12\$:	RET		:	

; Routine Size: 266 bytes, Routine Base: \$CODE\$ + 114B

```
1624 2150 1
1625 2151 1 routine ALLOCATE_TABLE: novalue =
1626 2152 2 begin
1627 2153 2
1628 2154 2 ++
1629 2155 2
1630 2156 2 Functional Description:
1631 2157 2     Initially allocate the usage table
1632 2158 2
1633 2159 2 Calling Sequence:
1634 2160 2
1635 2161 2     Normal
1636 2162 2
1637 2163 2 Input Parameters:
1638 2164 2
1639 2165 2     None
1640 2166 2
1641 2167 2 Implicit Inputs:
1642 2168 2
1643 2169 2     None
1644 2170 2
1645 2171 2 Output Parameters:
1646 2172 2
1647 2173 2     None
1648 2174 2
1649 2175 2 Implicit Outputs:
1650 2176 2
1651 2177 2     TABLE_SIZE           The size, in bytes, of the usage table
1652 2178 2     ENTRIES_IN_TABLE       The number of buckets in that table
1653 2179 2     USAGE_TABLE            Hash table into which entries are made
1654 2180 2
1655 2181 2 --
1656 2182 2
1657 2183 2
1658 2184 2 external routine
1659 2185 2
1660 2186 2     LIB$GET_VM: addressing_mode(general);      ! allocate virtual memory
1661 2187 2
1662 2188 2 local
1663 2189 2
1664 2190 2     STATUS;
1665 2191 2
1666 2192 2
1667 2193 2     Compute the size of the hash table
1668 2194 2
1669 2195 2
1670 2196 2     TABLE_SIZE = max(16*((512/DQF$C_LENGTH)*.QUOTA_EOF - 1), 16*511);
1671 2197 2     ENTRIES_IN_TABLE = .TABLE_SIZE/T6;
1672 2198 2
1673 2199 2
1674 2200 2     Allocate the table
1675 2201 2
1676 2202 2
1677 2203 2     STATUS = LIB$GET_VM(TABLE_SIZE, USAGE_TABLE);
1678 2204 2
1679 2205 2     if not .STATUS then
1680 2206 2         begin
```

```
: 1681      2207      3      USAGE TABLE = 0;  
: 1682      2208      3      RBLD_EXIT(RBLD$MEMALLOC, .STATUS);  
: 1683      2209      3      end;  
: 1684      2210      3  
: 1685      2211      3      ch$fill(0, .TABLE_SIZE, .USAGE_TABLE);  
: 1686      2212      3  
: 1687      2213      1 end;
```

007C 00000 ALLOCATE_TABLE:									
		56	00000000'	EF	9E	00002	WORD	Save R2,R3,R4,R5,R6	: 2151
	50	38	A6	04	78	00009	MOVAB	TABLE_SIZE, R6	: 2196
				50	D7	0000E	ASHL	#4, QOTA_EOF, R0	
			50	10	C4	00010	DECL	R0	
		00001FF0	8F	50	D1	00013	MULL2	#16, R0	
				05	18	0001A	CMPL	R0, #8176	
			50	8F	3C	0001C	BGEQ	1\$	
			66	50	D0	00021	MOVZWL	#8176, R0	
04	A6		66	10	C7	00024	MOVL	R0, TABLE_SIZE	: 2197
			08	A6	9F	00029	DIVL3	#16, TABLE_SIZE, ENTRIES_IN_TABLE	: 2203
				56	DD	0002C	PUSHAB	USAGE_TABLE	
		00000000G	00	02	FB	0002E	PUSHL	R6	
			12	50	E8	00035	CALLS	#2, LIB\$GET_VM	: 2205
			08	A6	D4	00038	BLBS	STATUS, 2\$: 2207
				50	DD	0003B	CLRL	USAGE_TABLE	: 2208
				8F	DD	0003D	PUSHL	STATUS	
		00000000G	00	02	FB	00043	PUSHL	#4522136	
66	00		6E	00	2C	0004A	CALLS	#2, LIB\$STOP	: 2211
			08	B6		0004F	MOVCS	#0, (SP), #0, TABLE_SIZE, @USAGE_TABLE	: 2213
				04		00051	RET		

; Routine Size: 82 bytes, Routine Base: \$CODE\$ + 1255

; 1688 2214 1

```
1690 2215 1
1691 2216 1 routine COUNT_QUOTA(UIC, USAGE, PRESCAN): novalue =
1692 2217 1 begin
1693 2218 1
1694 2219 1 ++
1695 2220 1
1696 2221 1 Functional Description:
1697 2222 1
1698 2223 1 Add an entry to the usage table. The usage table is a hash table
1699 2224 1 whose key is the UIC which is to be stored there. Overflow is handled
1700 2225 1 by chaining each bucket.
1701 2226 1
1702 2227 1 Calling Sequence:
1703 2228 1
1704 2229 1 COUNT_QUOTA(.UIC, .USAGE, .PRESCAN);
1705 2230 1
1706 2231 1 Input Parameters:
1707 2232 1
1708 2233 1 UIC The UIC of the file in question
1709 2234 1 USAGE The number of blocks used by that file
1710 2235 1 PRESCAN 1, if this entry exists in the old QUOTA.SYS
1711 2236 1
1712 2237 1 Implicit Inputs:
1713 2238 1
1714 2239 1 USAGE_TABLE The hash table itself
1715 2240 1
1716 2241 1 Output Parameters:
1717 2242 1
1718 2243 1 None
1719 2244 1
1720 2245 1 Implicit Outputs:
1721 2246 1
1722 2247 1 USAGE_TABLE The hash table itself
1723 2248 1
1724 2249 1 --
1725 2250 1
1726 2251 1 external routine
1727 2252 1
1728 2253 1 LIB$GET_VM: addressing_mode(general);
1729 2254 1
1730 2255 1 local
1731 2256 1
1732 2257 1 Q: ref block[4];
1733 2258 1
1734 2259 1 The hash function
1735 2260 1
1736 2261 1
1737 2262 1
1738 2263 1 macro
1739 2264 1
1740 2265 1 h(UIC) = UIC mod .ENTRIES_IN_TABLE %;
1741 2266 1
1742 2267 1
1743 2268 1 Hash on the UIC into the usage table
1744 2269 1
1745 2270 1
1746 2271 1 Q = USAGE_TABLE[ h(.UIC<0,$bitposition(UIC$V_FORMAT)>), 0,0,32,0 ];
```

```
1747 2272 2
1748 2273 2
1749 2274 2
1750 2275 2
1751 2276 2
1752 2277 2
1753 2278 2
1754 2279 2
1755 2280 2
1756 2281 2
1757 2282 2
1758 2283 2
1759 2284 2
1760 2285 4
1761 2286 4
1762 2287 4
1763 2288 4
1764 2289 4
1765 2290 4
1766 2291 4
1767 2292 4
1768 2293 4
1769 2294 3
1770 2295 4
1771 2296 4
1772 2297 4
1773 2298 4
1774 2299 4
1775 2300 4
1776 2301 4
1777 2302 3
1778 2303 4
1779 2304 4
1780 2305 4
1781 2306 4
1782 2307 4
1783 2308 3
1784 2309 3
1785 2310 3
1786 2311 3
1787 2312 3
1788 2313 3
1789 2314 3
1790 2315 3
1791 2316 2
1792 2317 2
1793 2318 1

Search the chain for the entry which matches our UIC

while SSS_NORMAL do
begin

Compare the UIC field of each chain member against our UIC

if not .Q[UTB_V_INUSE] then
begin
Q[UTB_L_UIC] = .UIC;
Q[UTB_L_USAGE] = .USAGE;
Q[UTB_V_PRESCAN] = .PRESCAN;
Q[UTB_V_INUSE] = 1;
exitloop;
end
else if .Q[UTB_L_UIC] eql .UIC then
begin
Q[UTB_L_USAGE] = .Q[UTB_L_USAGE] + .USAGE;
Q[UTB_V_PRESCAN] = .Q[UTB_V_PRESCAN] or .PRESCAN;
exitloop;
end
else if .Q[UTB_A_NEXT] eql 0 then
begin
LIB$GET_VM(uplit(16), Q[UTB_A_NEXT]);
ch$fill(0, 16, .Q[UTB_A_NEXT]);
end;
end;

Try the next entry in the chain

Q = .Q[UTB_A_NEXT];

end;

! end of routine COUNT_QUOTA
```

.PSECT SPLITS,NOWRT,NOEXE,2

00000010 0000C P.AAF: .LONG 16

.PSECT \$CODE\$,NOWRT,2


```
1796 2320 1
1797 2321 1 routine DELETE_TABLE: novalue =
1798 2322 2 begin
1799 2323 2
1800 2324 2 ++
1801 2325 2
1802 2326 2 Functional Description:
1803 2327 2
1804 2328 2 This routine deletes the usage table and returns the space to
1805 2329 2 the free storage pool.
1806 2330 2
1807 2331 2 Calling Sequence:
1808 2332 2
1809 2333 2 Standard
1810 2334 2
1811 2335 2 Input Parameters:
1812 2336 2
1813 2337 2 None
1814 2338 2
1815 2339 2 Implicit Inputs:
1816 2340 2
1817 2341 2 USAGE_TABLE The table to be disposed of
1818 2342 2
1819 2343 2 Output Parameters:
1820 2344 2
1821 2345 2 None
1822 2346 2
1823 2347 2 Implicit Outputs:
1824 2348 2
1825 2349 2 None
1826 2350 2
1827 2351 2 Routine Value:
1828 2352 2
1829 2353 2 None
1830 2354 2
1831 2355 2 --
1832 2356 2
1833 2357 2 external routine
1834 2358 2
1835 2359 2 LIB$FREE_VM: addressing_mode(general);
1836 2360 2
1837 2361 2 local
1838 2362 2
1839 2363 2 P: ref block[4],
1840 2364 2 Q,
1841 2365 2 R;
1842 2366 2
1843 2367 2
1844 2368 2 Check all the link pointers in the table
1845 2369 2
1846 2370 2
1847 2371 2 Q = USAGE_TABLE[0, UTB_A_NEXT];
1848 2372 2
1849 2373 2
1850 2374 2 Deallocate all the chained entries first
1851 2375 2
1852 2376 2
```

```
1853 2377 2 until .Q geqa .USAGE_TABLE+.TABLE_SIZE do
1854 2378 3 begin
1855 2379 4
1856 2380 5     P = ..Q;
1857 2381 6
1858 2382 7     until .P eql 0 do
1859 2383 8         begin
1860 2384 9
1861 2385 10             R = .P;
1862 2386 11             P = .P[UTB_A_NEXT];
1863 2387 12
1864 2388 13             LIB$FREE_VM(uplit(16), R);
1865 2389 14
1866 2390 15         end;
1867 2391 16
1868 2392 17         Q = .Q + 16;
1869 2393 18
1870 2394 19     end;
1871 2395 20
1872 2396 21     !
1873 2397 22     Deallocate the table itself
1874 2398 23     !
1875 2399 24
1876 2400 25     LIB$FREE_VM(TABLE_SIZE, USAGE_TABLE);
1877 2401 26
1878 2402 27 end; ! end of routine DELETE_TABLE
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

00000010 00010 P.AAG: .LONG 16

.PSECT \$CODE\$,NOWRT,2

003C 00000 DELETE_TABLE:

53	55	00000000G	00	9E	00002	.WORD	Save R2,R3,R4,R5	2321
50	54	000000000'	EF	9E	00009	MOVAB	LIB\$FREE_VM, R5	
	5E		04	C2	00010	MOVAB	USAGE_TABLE, R4	
	64		08	C1	00013	SUBL2	#4, SP	2371
	64	F8	A4	C1	00017	ADDL3	#8, USAGE_TABLE, Q	2377
	50		53	D1	0001C	ADDL3	TABLE_SIZE, USAGE_TABLE, R0	
			20	1E	0001F	CMPL	Q, R0	
	52		63	D0	00021	BGEQU	4\$	2380
			52	D5	00024	MOVL	(Q), P	2382
			14	13	00026	TSTL	P	
	6E		52	D0	00028	BEQL	3\$	2385
	52	08	A2	D0	0002B	MOVL	P, R	2386
			5E	DD	0002F	MOVL	8(P), P	2388
		000000000'	EF	9F	00031	PUSHL	SP	
	65		02	FB	00037	PUSHAB	P.AAG	
			E8	11	0003A	CALLS	#2, LIB\$FREE_VM	2382
	53		10	C0	0003C	BRB	2\$	2392
			D6	11	0003F	ADDL2	#16, Q	2377
						BRB	1\$	

```

N 9
16-Sep-1984 01:27:55 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:34 [MOUNT.SRC]REBUILD.B32;2

```

REB
V74

```

65          F8      54 DD 00041 4$: PUSHL R4
              A4 9F 00043      PUSHAB TABLE_SIZE
              02 FB 00046      CALLS #2, LIB$FREE_VM
              04 00049      RET

```

: 2400
:
:
:
:
:
: 2402

: 1879 2403 1



```
1881 2404 1 ROUTINE MARK_ALLOC (COUNT, LBN) : NOVALUE =
1882 2405 1
1883 2406 1 ++
1884 2407 1
1885 2408 1 Functional Description:
1886 2409 1
1887 2410 1 This routine marks the specified blocks as allocated in the allocation
1888 2411 1 bitmap.
1889 2412 1
1890 2413 1 Calling Sequence:
1891 2414 1 standard
1892 2415 1
1893 2416 1 Input Parameters:
1894 2417 1 none
1895 2418 1
1896 2419 1 Implicit Inputs:
1897 2420 1 none
1898 2421 1
1899 2422 1 Output Parameters:
1900 2423 1 none
1901 2424 1
1902 2425 1 Implicit Outputs:
1903 2426 1 none
1904 2427 1
1905 2428 1 Routines Called:
1906 2429 1 none
1907 2430 1
1908 2431 1 Routine Value:
1909 2432 1 none
1910 2433 1
1911 2434 1 Signals:
1912 2435 1 none
1913 2436 1
1914 2437 1 Side Effects:
1915 2438 1 none
1916 2439 1
1917 2440 1 --
1918 2441 1
1919 2442 2 BEGIN
1920 2443 2
1921 2444 2 LOCAL
1922 2445 2 CLUSTER_NUMBER, ! Cluster bit number
1923 2446 2 CLUSTER_COUNT, ! Count of clusters to mark allocated
1924 2447 2 J; ! Bitmap index
1925 2448 2
1926 2449 2 CLUSTER_COUNT = .COUNT<0,31> / .ALLOC_CLUSTER;
1927 2450 2 CLUSTER_NUMBER = .LBN<0,31> / .ALLOC_CLUSTER;
1928 2451 2 INCR J FROM .CLUSTER_NUMBER
1929 2452 2 TO MINU (.CLUSTER_NUMBER + .CLUSTER_COUNT, .ALLOCMAP_SIZE * 8) - 1
1930 2453 2 DO
1931 2454 3 BEGIN
1932 2455 3 IF NOT .ALLOCMAP[J]
1933 2456 3 THEN DUALLOC = 1
1934 2457 3 ELSE BLOCKS_AVAIL = .BLOCKS_AVAIL - .ALLOC_CLUSTER;
1935 2458 3 ALLOCMAP[J] = 0;
1936 2459 2 END;
1937 2460 1 END; ! end of routine MARK_ALLOC
```

```
001C 00000 MARK_ALLOC:
      54 00000000' EF 9E 00002 .WORD Save R2,R3,R4 : 2404
      53 08      A4 D0 00009 MOVAB ALLOCMAP, R4 :
      1F      00 EF 0000D MOVL ALLOC_CLUSTER, R3 : 2449
      51      53 C6 00013 EXTZV #0, #31, COUNT, CLUSTER_COUNT :
      1F      00 EF 00016 EXTZV #0, #31, LBN, CLUSTER_NUMBER : 2450
      50      53 C6 0001C DIVL2 R3, CLUSTER_NUMBER :
      50      51 C1 0001F ADDL3 CLUSTER_COUNT, CLUSTER_NUMBER, R2 : 2452
      51      04 A4 03 78 00023 ASHL #3, ALLOCMAP_SIZE, R1 :
      51      52 D1 00028 CMPL R2, R1 :
      52      03 18 0002B BLEQU 1$ :
      51      51 D0 0002D MOVL R1, R2 :
      50      14 11 00032 1$: DECL J : 2455
      50      50 E0 00034 2$: BRB 5$ :
      01      04 11 0003D BRB 4$ : 2456
      53      53 C2 0003F 3$: SUBL2 R3, BLOCKS_AVAIL : 2457
      50      50 00 00 00 50 E5 00043 4$: BBCC J, @ALLOCMP, 5$ : 2458
      52      52 F2 00048 5$: AOBLSS R2, J, 2$ : 2451
      04 0004C RET : 2460
```

; Routine Size: 77 bytes, Routine Base: \$CODE\$ + 1365

```
: 1939 2461 1 ROUTINE VERIFY_HEADER (HEADER, FILE_ID) =
: 1940 2462 1
: 1941 2463 1 :++
: 1942 2464 1
: 1943 2465 1 FUNCTIONAL DESCRIPTION:
: 1944 2466 1
: 1945 2467 1 This routine verifies that the block given it is in fact a
: 1946 2468 1 file header. If file number and/or file sequence number are also
: 1947 2469 1 supplied, they are checked as well.
: 1948 2470 1
: 1949 2471 1 Calling Sequence:
: 1950 2472 1 standard
: 1951 2473 1
: 1952 2474 1 Input Parameters:
: 1953 2475 1 none
: 1954 2476 1
: 1955 2477 1 Implicit Inputs:
: 1956 2478 1 none
: 1957 2479 1
: 1958 2480 1 Output Parameters:
: 1959 2481 1 none
: 1960 2482 1
: 1961 2483 1 Implicit Outputs:
: 1962 2484 1 none
: 1963 2485 1
: 1964 2486 1 Routines Called:
: 1965 2487 1 none
: 1966 2488 1
: 1967 2489 1 Routine Value:
: 1968 2490 1 none
: 1969 2491 1
: 1970 2492 1 Signals:
: 1971 2493 1 none
: 1972 2494 1
: 1973 2495 1 Side Effects:
: 1974 2496 1 none
: 1975 2497 1
: 1976 2498 1 --
: 1977 2499 1
: 1978 2500 2 BEGIN
: 1979 2501 2
: 1980 2502 2 MAP
: 1981 2503 2 HEADER : REF BBLOCK, ! file header arg
: 1982 2504 2 FILE_ID : REF BBLOCK; ! file ID arg
: 1983 2505 2
: 1984 2506 2 LOCAL
: 1985 2507 2 MAP_AREA : REF BBLOCK; ! pointer to header map area
: 1986 2508 2
: 1987 2509 2 EXTERNAL ROUTINE
: 1988 2510 2 CHECKSUM; ! compute file header checksum
: 1989 2511 2
: 1990 2512 2
: 1991 2513 2 ! First check the structure level.
: 1992 2514 2 !
: 1993 2515 2
: 1994 2516 2 IF .HEADER[FH2$B_STRUCLEV] NEQ 2
: 1995 2517 2 THEN RETURN 0;
```

```
: 1996 2518 2
: 1997 2519 2 ! Check the area offsets and the retrieval pointer use counts for
: 1998 2520 2 consistency.
: 1999 2521 2
: 2000 2522 2
: 2001 2523 2 IF .HEADER[FH2$B_IDOFFSET] LSSU $BYTEOFFSET (FH2$L_HIGHWATER)/2
: 2002 2524 2 OR .HEADER[FH2$B_MPOFFSET] LSSU .HEADER[FH2$B_IDOFFSET]
: 2003 2525 2 OR .HEADER[FH2$B_ACOFFSET] LSSU .HEADER[FH2$B_MPOFFSET]
: 2004 2526 2 OR .HEADER[FH2$B_RSOFFSET] LSSU .HEADER[FH2$B_ACOFFSET]
: 2005 2527 2 OR .HEADER[FH2$B_MAP_INUSE] GTRU .HEADER[FH2$B_ACOFFSET] - .HEADER[FH2$B_MPOFFSET]
: 2006 2528 2 THEN RETURN 0;
: 2007 2529 2
: 2008 2530 2 ! At this point, we have verified that the block at least once was a
: 2009 2531 2 valid file header.
: 2010 2532 2
: 2011 2533 2 ! Look at the file number in the header. If zero, this is a
: 2012 2534 2 deleted header.
: 2013 2535 2
: 2014 2536 2
: 2015 2537 2 IF .HEADER[FH2$W_FID_NUM] EQL 0
: 2016 2538 2 AND .HEADER[FH2$B_FID_NMX] EQL 0
: 2017 2539 2 THEN RETURN 2;
: 2018 2540 2
: 2019 2541 2 ! Now compute the header checksum.
: 2020 2542 2
: 2021 2543 2
: 2022 2544 2 IF NOT CHECKSUM (.HEADER)
: 2023 2545 2 THEN RETURN 2;
: 2024 2546 2
: 2025 2547 2 ! Check file number and file sequence number.
: 2026 2548 2
: 2027 2549 2
: 2028 2550 2 IF .HEADER[FH2$W_FID_NUM] NEQ .FILE_ID[FID$W_NUM]
: 2029 2551 2 OR .HEADER[FH2$B_FID_NMX] NEQ .FILE_ID[FID$B_NMX]
: 2030 2552 2 THEN RETURN 2;
: 2031 2553 2
: 2032 2554 2 IF .HEADER[FH2$W_FID_SEQ] NEQ .FILE_ID[FID$W_SEQ]
: 2033 2555 2 THEN RETURN 2;
: 2034 2556 2
: 2035 2557 2 ! Header is ok.
: 2036 2558 2
: 2037 2559 2
: 2038 2560 2 RETURN 1;
: 2039 2561 2
: 2040 2562 1 END;
```

! end of routine VERIFY_HEADER

0004 00000 VERIFY_HEADER:

52	04	AC	D0	00002	.WORD	Save R2
02	07	A2	91	00006	MOVL	HEADER, R2
		62	12	0000A	CMPB	7(R2), #2
26		62	91	0000C	BNEQ	4\$
		5D	1F	0000F	CMPB	(R2), #38
					BLSSU	4\$

```
: 2461
: 2516
:
: 2523
:
```

REBUILD
V04-000

F 10
16-Sep-1984 01:27:55 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:34 [MOUNT.SRC]REBUILD.B32;2

Page 71
(10)

REB
V04

		62	01	A2	91	00011	CMPB	1(R2), (R2)	2524
				57	1F	00015	BLSSU	4\$	
	01	A2	02	A2	91	00017	CMPB	2(R2), 1(R2)	2525
				50	1F	0001C	BLSSU	4\$	
	02	A2	03	A2	91	0001E	CMPB	3(R2), 2(R2)	2526
				49	1F	00023	BLSSU	4\$	
		50	02	A2	9A	00025	MOVZBL	2(R2), R0	2527
		51	01	A2	9A	00029	MOVZBL	1(R2), R1	
		50		51	C2	0002D	SUBL2	R1, R0	
50				00	ED	00030	CMPZV	#0, #8, 58(R2), R0	
	3A	A2		36	1A	00036	BGTRU	4\$	
			08	A2	B5	00038	TSTW	8(R2)	2537
				05	12	0003B	BNEQ	1\$	
			0D	A2	95	0003D	TSTB	13(R2)	2538
				24	13	00040	BEQL	2\$	
				52	DD	00042	PUSHL	R2	2544
	00000000G	00		01	FB	00044	CALLS	#1, CHECKSUM	
		18		50	E9	0004B	BLBC	R0, 2\$	
		50	08	AC	D0	0004E	MOVL	FILE_ID, R0	2550
		60	08	A2	B1	00052	CMPW	8(R2), (R0)	
				0E	12	00056	BNEQ	2\$	
	05	A0	0D	A2	91	00058	CMPB	13(R2), 5(R0)	2551
				07	12	0005D	BNEQ	2\$	
	02	A0	0A	A2	B1	0005F	CMPW	10(R2), 2(R0)	2554
				04	13	00064	BEQL	3\$	
		50		02	D0	00066	MOVL	#2, R0	2555
					04	00069	RET		
		50		01	D0	0006A	MOVL	#1, R0	2560
					04	0006D	RET		
				50	D4	0006E	CLRL	R0	2562
					04	00070	RET		

; Routine Size: 113 bytes, Routine Base: \$CODE\$ + 13B2

```
2042 2563 1 ROUTINE READ_HOMEBLOCK (BUFFER, RVN) : NOVALUE =
2043 2564 1
2044 2565 1 ++
2045 2566 1
2046 2567 1 Functional Description:
2047 2568 1
2048 2569 1 This routine reads the first good home block of the currently open
2049 2570 1 index file into the buffer supplied.
2050 2571 1
2051 2572 1 Calling Sequence:
2052 2573 1 standard
2053 2574 1
2054 2575 1 Input Parameters:
2055 2576 1 none
2056 2577 1
2057 2578 1 Implicit Inputs:
2058 2579 1 none
2059 2580 1
2060 2581 1 Output Parameters:
2061 2582 1 none
2062 2583 1
2063 2584 1 Implicit Outputs:
2064 2585 1 none
2065 2586 1
2066 2587 1 Routines Called:
2067 2588 1 none
2068 2589 1
2069 2590 1 Routine Value:
2070 2591 1 none
2071 2592 1
2072 2593 1 Signals:
2073 2594 1 none
2074 2595 1
2075 2596 1 Side Effects:
2076 2597 1 none
2077 2598 1
2078 2599 1 --
2079 2600 1
2080 2601 2 BEGIN
2081 2602 2
2082 2603 2 MAP
2083 2604 2 BUFFER : REF BBLOCK; ! block buffer arg
2084 2605 2
2085 2606 2 LOCAL
2086 2607 2 STATUS, ! general status value
2087 2608 2 OLD_STATUS; ! save status for error message
2088 2609 2
2089 2610 2
2090 2611 2 ! We keep reading until we get a block that reads without errors and looks
2091 2612 2 ! like a " " block. Track any error status for the eventual error message.
2092 2613 2
2093 2614 2
2094 2615 2 OLD STATUS = $$$ ABORT;
2095 2616 2 INCR VBN FROM 2 TO 100
2096 2617 2 DO
2097 2618 2 BEGIN
2098 P 2619 2 STATUS = DO_IO (CHAN = .CHANNEL,
```

; R

```
2099      P 2620      3      FUNC = IOS_READVELK,  
2100      P 2621      3      IOSB = IO_STATUS,  
2101      P 2622      3      P1 = .BUFFER,  
2102      P 2623      3      P2 = $12,  
2103      P 2624      3      P3 = .VBN  
2104      P 2625      3      );  
2105      P 2626      3      IF .STATUS THEN STATUS = .IO_STATUS[0];  
2106      P 2627      3  
2107      P 2628      3      IF NOT .STATUS  
2108      P 2629      3      THEN  
2109      P 2630      3          OLD_STATUS = .STATUS  
2110      P 2631      3      ELSE  
2111      P 2632      3          IF CHECK_HOMEBLOCK (.BUFFER, .VBN)  
2112      P 2633      3          THEN RETURN;  
2113      P 2634      3      END;  
2114      P 2635      3          ! end of loop  
2115      P 2636      3      RBLD_EXIT (RBLD$_HOMEBLOCK, .RVN, .OLD_STATUS);  
2116      P 2637      3  
2117      P 2638      1  END;  
2117      P 2638      1          ! end of routine READ_HOMEBLOCK
```

```
003C 00000 READ_HOMEBLOCK:  
55 00000000' EF 9E 00002 .WORD Save R2,R3,R4,R5 : 2563  
54 2C D0 00009 MOVAB IO_STATUS, R5 : 2615  
52 02 D0 0000C MOVL #4, OLD_STATUS : 2625  
7E 0200 7E 7C 0000F 1$: CLRG -(SP)  
04 AC DD 0001A CLRL -(SP)  
7E FA 52 DD 00013 PUSHL VBN  
00 00 8F 3C 00015 MOVZWL #512, -(SP)  
53 06 AC DD 0001A PUSHL BUFFER  
53 65 3C 00036 CLRG -(SP)  
05 53 E8 00039 PUSHL R5  
54 53 D0 0003C 31 DD 00021 PUSHL #49  
0F 11 0003F A5 3C 00023 MOVZWL CHANNEL, -(SP)  
52 DD 00041 1A DD 00027 PUSHL #26  
AC DD 00043 0C FB 00029 CALLS #12, COMMON_IO  
02 FB 00046 MOVL R0, STATUS : 2626  
50 E8 0004D 53 E9 00033 BLBC STATUS, 2$ : 2628  
8F F3 00050 65 3C 00036 MOVZWL IO_STATUS, STATUS : 2630  
54 DD 00058 53 E8 00039 BLBS STATUS, 3$  
AC DD 0005A 53 D0 0003C 2$: MOVL STATUS, OLD_STATUS : 2632  
8F DD 0005D 0F 11 0003F BRB 4$  
03 FB 00063 52 DD 00041 3$: PUSHL VBN  
04 0006A 5$: PUSHL BUFFER : 2638  
EF 02 FB 00046 CALLS #2, CHECK_HOMEBLOCK  
1A 50 E8 0004D BLBS R0, 5$  
52 00000064 8F F3 00050 4$: AOBLEQ #100, VBN, 1$ : 2616  
08 AC DD 0005A PUSHL OLD_STATUS : 2636  
00 004500A0 8F DD 0005D PUSHL RVN  
00 03 FB 00063 CALLS #3, LIB$STOP  
04 0006A 5$: RET
```

REBUILD
V04-000

1-10
16-Sep-1984 01:27:55
14-Sep-1984 12:45:34

VAX-11 Bliss-32 V4.0-742
[MOUNT.SRC]REBUILD.B32;2

Page 74
(11)

REB
V04

; Routine Size: 107 bytes, Routine Base: SCODES + 1423

: R

```
2119 2639 1 ROUTINE CHECK_HOMEBLOCK (HOME_BLOCK, VBN) =
2120 2640 1
2121 2641 1 ++
2122 2642 1
2123 2643 1 FUNCTIONAL DESCRIPTION:
2124 2644 1
2125 2645 1 This routine verifies whether the given block is a Files-11 Structure
2126 2646 1 Level 2 home block.
2127 2647 1
2128 2648 1 Calling Sequence:
2129 2649 1 standard
2130 2650 1
2131 2651 1 Input Parameters:
2132 2652 1 none
2133 2653 1
2134 2654 1 Implicit Inputs:
2135 2655 1 none
2136 2656 1
2137 2657 1 Output Parameters:
2138 2658 1 none
2139 2659 1
2140 2660 1 Implicit Outputs:
2141 2661 1 none
2142 2662 1
2143 2663 1 Routines Called:
2144 2664 1 none
2145 2665 1
2146 2666 1 Routine Value:
2147 2667 1 none
2148 2668 1
2149 2669 1 Signals:
2150 2670 1 none
2151 2671 1
2152 2672 1 Side Effects:
2153 2673 1 none
2154 2674 1
2155 2675 1 --
2156 2676 1
2157 2677 2 BEGIN
2158 2678 2
2159 2679 2 MAP
2160 2680 2 HOME_BLOCK : REF BBLOCK; ! home block buffer
2161 2681 2
2162 2682 2 EXTERNAL ROUTINE
2163 2683 2 CHECKSUM2; ! compute home block checksums
2164 2684 2
2165 2685 2
2166 2686 2 ! Check the required non-zero fields and compute the checksums.
2167 2687 2
2168 2688 2
2169 2689 2 IF NOT (
2170 2690 2 .HOME_BLOCK[HM2$W_HOMEVBN] EQL .VBN
2171 2691 2 AND .HOME_BLOCK[HM2$W_ALTDIXLBN] NEQ 0
2172 2692 2 AND .HOME_BLOCK[HM2$W_CLUSTER] NEQ 0
2173 2693 2 AND .HOME_BLOCK[HM2$W_HOMEVBN] NEQ 0
2174 2694 2 AND .HOME_BLOCK[HM2$W_ALHOMEVBN] NEQ 0
2175 2695 2 AND .HOME_BLOCK[HM2$W_ALTDIXVBN] NEQ 0
```

```
: 2176      2696 3      AND .HOME_BLOCK[HM2$W-IBMAPVBN] NEQ 0
: 2177      2697 3      AND .HOME_BLOCK[HM2$W-IBMAPLBN] NEQ 0
: 2178      2698 3      AND .HOME_BLOCK[HM2$W-MAXFILES] NEQ 0
: 2179      2699 3      AND .HOME_BLOCK[HM2$W-IBMAPSIZE] NEQ 0
: 2180      2700 3      AND .HOME_BLOCK[HM2$W-RESFILES] NEQ 0
: 2181      2701 3      AND CHECKSUM2 (.HOME_BLOCK, $BYTEOFFSET (HM2$W-CHECKSUM1))
: 2182      2702 3      AND CHECKSUM2 (.HOME_BLOCK, $BYTEOFFSET (HM2$W-CHECKSUM2))
: 2183      2703 3      )
: 2184      2704 2      THEN RETURN 0;
: 2185      2705 2
: 2186      2706 2      RETURN 1;
: 2187      2707 2
: 2188      2708 1      END;

! end of routine CHECK_HOMEBLOCK
```

```
                                .EXTRN  CHECKSUM2
                                000C 00000 CHECK_HOMEBLOCK:
                                .WORD  Save R2,R3
                                MOVAB  CHECKSUM2, R3
                                MOVL   HOME_BLOCK, R2
                                CMPZV  #0, #16, 16(R2), VBN
                                BNEQ   1$
                                TSTL   8(R2)
                                BEQL   1$
                                TSTW   14(R2)
                                BEQL   1$
                                TSTW   16(R2)
                                BEQL   1$
                                TSTW   18(R2)
                                BEQL   1$
                                TSTW   20(R2)
                                BEQL   1$
                                TSTW   22(R2)
                                BEQL   1$
                                TSTL   24(R2)
                                BEQL   1$
                                TSTL   28(R2)
                                BEQL   1$
                                TSTW   32(R2)
                                BEQL   1$
                                TSTW   34(R2)
                                BEQL   1$
                                PUSHL  #58
                                PUSHL  R2
                                CALLS  #2, CHECKSUM2
                                BLBC   R0, 1$
                                MOVZWL #510, -(SP)
                                PUSHL  R2
                                CALLS  #2, CHECKSUM2
                                BLBC   R0, 1$
                                MOVL   #1, R0
                                RET
                                CLRL   R0
                                RET

08  AC      10  A2      53 00000000G 00 9E 00002  .WORD  Save R2,R3
                                52      04  AC  D0 00009  MOVAB  CHECKSUM2, R3
                                10      00  ED 0000D  MOVL   HOME_BLOCK, R2
                                08      4D  12 00014  CMPZV  #0, #16, 16(R2), VBN
                                48      A2  D5 00016  BNEQ   1$
                                0E      A2  B5 0001B  TSTL   8(R2)
                                43      13 0001E  BEQL   1$
                                10      A2  B5 00020  TSTW   14(R2)
                                3E      13 00023  BEQL   1$
                                12      A2  B5 00025  TSTW   16(R2)
                                39      13 00028  BEQL   1$
                                14      A2  B5 0002A  TSTW   18(R2)
                                34      13 0002D  BEQL   1$
                                16      A2  B5 0002F  TSTW   20(R2)
                                2F      13 00032  BEQL   1$
                                18      A2  D5 00034  TSTW   22(R2)
                                2A      13 00037  BEQL   1$
                                1C      A2  D5 00039  TSTL   24(R2)
                                25      13 0003C  BEQL   1$
                                20      A2  B5 0003E  TSTL   28(R2)
                                20      13 00041  BEQL   1$
                                22      A2  B5 00043  TSTW   32(R2)
                                1B      13 00046  BEQL   1$
                                3A      DD 00048  TSTW   34(R2)
                                52      DD 0004A  BEQL   1$
                                63      02  FB 0004C  PUSHL  #58
                                11      50  E9 0004F  PUSHL  R2
                                7E      8F  3C 00052  CALLS  #2, CHECKSUM2
                                52      DD 00057  BLBC   R0, 1$
                                63      02  FB 00059  MOVZWL #510, -(SP)
                                04      50  E9 0005C  PUSHL  R2
                                50      01  D0 0005F  CALLS  #2, CHECKSUM2
                                04      04  00062  BLBC   R0, 1$
                                50      D4  00063  MOVL   #1, R0
                                04      04  00065  RET
                                1$:
                                04      04  00065  CLRL   R0
                                RET

                                : 2639
                                : 2690
                                : 2691
                                : 2692
                                : 2693
                                : 2694
                                : 2695
                                : 2696
                                : 2697
                                : 2698
                                : 2699
                                : 2700
                                : 2701
                                : 2702
                                : 2706
                                : 2708
```

REBUILD
V04-000

L 10
16-Sep-1984 01:27:55
14-Sep-1984 12:45:34

VAX-11 Bliss-32 V4.0-742
[MOUNT.SRC]REBUILD.B32;2

Page 77
(12)

```
; Routine Size: 102 bytes,    Routine Base: $CODES + 148E
```

REB
V04

```
2190 2709 1 ROUTINE RBLD_HANDLER (SIGNAL_VEC, MECHANISM) =
2191 2710 1
2192 2711 1 ++
2193 2712 1
2194 2713 1 Functional Description:
2195 2714 1
2196 2715 1 This routine is the main condition handler for the DISKQUOTA utility.
2197 2716 1 It receives a signal which is either an internal error code or a
2198 2717 1 standard system status. If the former, the appropriate message is
2199 2718 1 formatted and printed. For the latter, the condition is simply
2200 2719 1 resigalled.
2201 2720 1
2202 2721 1 Calling Sequence:
2203 2722 1 standard
2204 2723 1
2205 2724 1 Input Parameters:
2206 2725 1 none
2207 2726 1
2208 2727 1 Implicit Inputs:
2209 2728 1 none
2210 2729 1
2211 2730 1 Output Parameters:
2212 2731 1 none
2213 2732 1
2214 2733 1 Implicit Outputs:
2215 2734 1 none
2216 2735 1
2217 2736 1 Routines Called:
2218 2737 1 none
2219 2738 1
2220 2739 1 Routine Value:
2221 2740 1 none
2222 2741 1
2223 2742 1 Signals:
2224 2743 1 none
2225 2744 1
2226 2745 1 Side Effects:
2227 2746 1 none
2228 2747 1
2229 2748 1 --
2230 2749 1
2231 2750 2 BEGIN
2232 2751 2
2233 2752 2 MAP
2234 2753 2 SIGNAL_VEC : REF BBLOCK, ! signal vector arg
2235 2754 2 MECHANISM : REF BBLOCK; ! mechanism vector arg
2236 2755 2
2237 2756 2 LOCAL
2238 2757 2 FORMAT_DESC : VECTOR [2], ! string descriptor for message format
2239 2758 2 P : REF VECTOR [,BYTE], ! string pointer
2240 2759 2 ERR_CODE : BBLOCK [4]; ! error status code
2241 2760 2
2242 2761 2 EXTERNAL ROUTINE
2243 2762 2 LIB$PUT_OUTPUT : ADDRESSING_MODE (GENERAL),
2244 2763 2 LIB$FREE_VM : ADDRESSING_MODE (GENERAL);
2245 2764 2
2246 2765 2
```

```
2247 2766 2 ! Get the signal code. If it is one of ours, get the message string and
2248 2767 2 ! do formatting as necessary.
2249 2768 2 !
2250 2769 2 !
2251 2770 2 ERR_CODE = .SIGNAL_VEC[CHFSL_SIG_NAME];
2252 2771 2 IF .ERR_CODE[STSSV_FAC_NO] EQL FAC_CODE
2253 2772 2 THEN
2254 2773 2 BEGIN
2255 2774 2 ERR_CODE = .ERR_CODE[STSSV_MSG_NO];
2256 2775 2 P = .MESSAGE_TABLE[ERR_CODE];
2257 2776 2 FORMAT_DESC[0] = .P[1];
2258 2777 2 FORMAT_DESC[1] = .P + 2;
2259 2778 2 OUTPUT_DESC[0] = OUTPUT_LENGTH;
2260 2779 2 OUTPUT_DESC[1] = OUTPUT_LINE;
2261 2780 2
2262 P 2781 2 $FAOL (CTRSTR = FORMAT_DESC[0],
2263 P 2782 2 OUTLEN = OUTPUT_DESC[0],
2264 P 2783 2 OUTBUF = OUTPUT_DESC[0],
2265 P 2784 2 PRMLST = SIGNAL_VEC[CHFSL_SIG_ARG1]
2266 2785 2 );
2267 2786 2 LIB$PUT_OUTPUT (OUTPUT_DESC);
2268 2787 2
2269 2788 2 ! If there is a signal argument remaining, it is a system error status.
2270 2789 2 ! Convert its severity to warning and signal it.
2271 2790 2 !
2272 2791 2
2273 2792 2 ERR_CODE = 0;
2274 2793 2 IF .SIGNAL_VEC[CHFSL_SIG_ARGS] GTRU .P[0] + 3
2275 2794 2 THEN
2276 2795 2 BEGIN
2277 2796 2 ERR_CODE = .VECTOR [SIGNAL_VEC[CHFSL_SIG_ARG1], .P[0]];
2278 2797 2 END;
2279 2798 2
2280 2799 2 END;
2281 2800 2 IF .ERR_CODE NEQ 0
2282 2801 2 AND .ERR_CODE NEQ SS$_UNWIND
2283 2802 2 THEN
2284 2803 2 BEGIN
2285 2804 2 ERR_CODE[STSSV_SEVERITY] = STSSK_WARNING;
2286 2805 2 SIGNAL (.ERR_CODE);
2287 2806 2 END;
2288 2807 2
2289 2808 2 MECHANISM[CHFSL_MCH_SAVRO] = 1;
2290 2809 2 IF .BBLOCK [SIGNAL_VEC[CHFSL_SIG_NAME], STSSV_SEVERITY] EQL STSSK_SEVERE
2291 2810 2 THEN
2292 2811 2 BEGIN
2293 2812 2 ! IF .CLEANUP_FLAGS[CLF_UNLOCK]
2294 2813 2 ! THEN
2295 2814 2 BEGIN
2296 2815 2 CH$FILL (0, FIBSC_LENGTH, QUOTA_FIB);
2297 2816 2 QUOTA_FIB[FIBSW_CNTRLFUNC] = FIBSC_UNLK_VOL;
2298 P 2817 2 DO_IO (CHAN = .CHANNEL,
2299 P 2818 2 FUNC = IOS_ACPCONTROL,
2300 P 2819 2 P1 = QFIB_DESC
2301 2820 2 );
2302 2821 2 ! CLEANUP_FLAGS[CLF_UNLOCK] = 0;
2303 2822 2 END;
```

```
2304 2823 3
2305 2824 3
2306 2825 3
2307 2826 3
2308 2827 3
2309 2828 3
2310 2829 4
2311 2830 4
2312 2831 4
2313 2832 3
2314 2833 3
2315 2834 3
2316 2835 3
2317 2836 4
2318 2837 4
2319 2838 4
2320 2839 3
2321 2840 3
2322 2841 3
2323 2842 3
2324 2843 4
2325 2844 4
2326 2845 4
2327 2846 3
2328 2847 3
2329 2848 3
2330 2849 3
2331 2850 4
2332 2851 4
2333 2852 4
2334 2853 3
2335 2854 3
2336 2855 3
2337 2856 3
2338 2857 4
2339 2858 4
2340 2859 4
2341 2860 3
2342 2861 3
2343 2862 3
2344 2863 3
2345 2864 3
2346 2865 4
2347 2866 4
2348 2867 4
2349 2868 3
2350 2869 3
2351 P 2870 3
2352 2871 3
2353 2872 3
2354 2873 3
2355 2874 3
2356 2875 3
2357 2876 3
2358 2877 3
2359 2878 3
2360 2879 2

IF .USAGE_TABLE NEQ 0
THEN DELETE_TABLE ();

IF .BUFFER NEQ 0
THEN
BEGIN
LIB$FREE_VM (UPLIT (BLOCK_FACTOR*512), BUFFER);
BUFFER = 0;
END;

IF .EOF NEQ 0
THEN
BEGIN
LIB$FREE_VM (DYN_SIZE, EOF);
EOF = 0;
END;

IF .IFILEMAP NEQ 0
THEN
BEGIN
LIB$FREE_VM (IFILEMAP_SIZE, IFILEMAP);
IFILEMAP = 0;
END;

IF .ALLOCMAP NEQ 0
THEN
BEGIN
LIB$FREE_VM (ALLOCMAP_SIZE, ALLOCMAP);
ALLOCMAP = 0;
END;

IF .OLD_ALLOCMAP NEQ 0
THEN
BEGIN
LIB$FREE_VM (UPLIT (512), OLD_ALLOCMAP);
OLD_ALLOCMAP = 0;
END;

IF .ERASE_CHANNEL NEQ 0
THEN
BEGIN
$DASSGN (CHAN=.ERASE_CHANNEL);
ERASE_CHANNEL = 0;
END;

DO_IO (CHAN = .CHANNEL,
FUNC = IOS_DEACCESS);

! Cancel the exit handler.
!
$CANEXH (DESBK=EXIT_HNDLR_DESC);

$UNWIND ();
END;
```

```
: 2361
: 2362
: 2363
: 2364

2880 2
2881 2 RETURN SSS_CONTINUE;
2882 2
2883 1 END;
```

! end of routine RBLD_HANDLER

```
.PSECT $SPLITS,NOWRT,NOEXE,2

00008000 00014 P.AAH: .LONG 32768
00000200 00018 P.AAI: .LONG 512

.EXTRN LIB$PUT_OUTPUT, SYSS$FAOL
.EXTRN SYSS$DASSGN, SYSS$UNWIND

.PSECT $CODE$,NOWRT,2

01FC 00000 RBLD_HANDLER:
      .WORD Save R2,R3,R4,R5,R6,R7,R8
      MOVAB COMMON IO, R8
      MOVAB LIB$FREE_VM, R7
      MOVAB OUTPUT_DESC, R6
      SUBL2 #8, SP
      MOVL SIGNAL_VEC, R4
      MOVL 4(R4), ERR_CODE
      CMPZV #16, #12, ERR_CODE, #69
      BNEQ 1$
      EXTZV #3, #13, ERR_CODE, ERR_CODE
      MOVL MESSAGE_TABLE[ERR_CODE], P
      MOVZBL 1(P), FORMAT_DESC
      MOVAB 2(R3), FORMAT_DESC+4
      MOVZBL #132, OUTPUT_DESC
      MOVAB OUTPUT_LINE, OUTPUT_DESC+4
      MOVL 8(R4), R5
      PUSHL R5
      PUSHL R6
      PUSHL R6
      PUSHAB FORMAT_DESC
      CALLS #4, SYSS$FAOL
      PUSHL R6
      CALLS #1, LIB$PUT_OUTPUT
      CLRL ERR_CODE
      MOVZBL (P), R0
      MOVAB 3(R0), R1
      CMPL (R4), R1
      BLEQU 1$
      MOVL (R5)[R0], ERR_CODE
      TSTL ERR_CODE
      BEQL 2$
      CMPL ERR_CODE, #2336
      BEQL 2$
      BICB2 #7, ERR_CODE
      PUSHL ERR_CODE
      CALLS #1, LIB$SIGNAL
      MOVL MECHANISM, R0
      MOVL #1, 12(R0)
      CMPZV #0, #3, 4(R4), #4

      2709
      2770
      2771
      2774
      2775
      2776
      2777
      2778
      2779
      2785
      2786
      2792
      2793
      2796
      2800
      2801
      2804
      2805
      2808
      2809
```

00000045	8F	52	58	00000000G	00	9E	00002	
			57	00000000G	00	9E	00009	
			56	00000000G	EF	9E	00010	
			5E		08	C2	00017	
			54	04	AC	D0	0001A	
			52	04	A4	D0	0001E	
			0C		10	ED	00022	
					4F	12	0002B	
			0D		03	EF	0002D	
			53	00000000G	EF	42	D0	00032
			6E	01	A3	9A	0003A	
	04		AE	02	A3	9E	0003E	
			66	84	8F	9A	00043	
	04		A6	FF7C	C6	9E	00047	
			55	08	A4	9E	0004D	
					55	DD	00051	
					56	DD	00053	
					56	DD	00055	
				0C	AE	9F	00057	
					04	FB	0005A	
					56	DD	00061	
					01	FB	00063	
					52	D4	0006A	
			50		63	9A	0006C	
			51	03	A0	9E	0006F	
			51		64	D1	00073	
					04	1B	00076	
			52		6540	D0	00078	
					52	D5	0007C	1\$:
					15	13	0007E	
					52	D1	00080	
					0C	13	00087	
			52		07	8A	00089	
					52	DD	0008C	
					01	FB	0008E	
				08	AC	D0	00095	2\$:
					01	D0	00099	
					00	ED	0009D	
04	04	A4		0C	03			

0040	8F	00	6E	03 13 000A3	BEQL	3\$:	
				00DA 31 000A5	BRW	11\$:	
				00 2C 000A8	3\$: MOVCS	#0, (SP), #0, #64, QUOTA_FIB	:	2815
			40	A6 000AF			:	
	56	A6		08 B0 000B1	MOVW	#8, QUOTA_FIB+22	:	2816
				7E 7C 000B5	CLRQ	-(SP)	:	2820
				7E 7C 000B7	CLRQ	-(SP)	:	
				7E D4 000B9	CLRL	-(SP)	:	
		00AC		C6 9F 000BB	PUSHAB	QFIB_DESC	:	
				7E 7C 000BF	CLRQ	-(SP)	:	
	7E			38 7D 000C1	MOVQ	#56, -(SP)	:	
	7E	FF6E		C6 3C 000C4	MOVZWL	CHANNEL, -(SP)	:	
				1A DD 000C9	PUSHL	#26	:	
	68			0C FB 000CB	CALLS	#12, COMMON_IO	:	
		00A0		C6 D5 000CE	TSTL	USAGE_TABLE	:	2824
				05 13 000D2	BEQL	4\$:	
	FD4E	CF		00 FB 000D4	CALLS	#0, DELETE_TABLE	:	2825
		FF4C		C6 D5 000D9	4\$: TSTL	BUFFER	:	2827
				11 13 000DD	BEQL	5\$:	
		FF4C		C6 9F 000DF	PUSHAB	BUFFER	:	2830
		00000000		EF 9F 000E3	PUSHAB	P.AAH	:	
	67			02 FB 000E9	CALLS	#2, LIB\$FREE_VM	:	
		FF4C		C6 D4 000EC	CLRL	BUFFER	:	2831
		0094		C6 D5 000F0	5\$: TSTL	EOF	:	2834
				0F 13 000F4	BEQL	6\$:	
		0094		C6 9F 000F6	PUSHAB	EOF	:	2837
		0080		C6 9F 000FA	PUSHAB	DYN_SIZE	:	
	67			02 FB 000FE	CALLS	#2, LIB\$FREE_VM	:	
		0094		C6 D4 00101	CLRL	EOF	:	2838
		FF50		C6 D5 00105	6\$: TSTL	IFILEMAP	:	2841
				0F 13 00109	BEQL	7\$:	
		FF50		C6 9F 0010B	PUSHAB	IFILEMAP	:	2844
		FF54		C6 9F 0010F	PUSHAB	IFILEMAP_SIZE	:	
	67			02 FB 00113	CALLS	#2, LIB\$FREE_VM	:	
		FF50		C6 D4 00116	CLRL	IFILEMAP	:	2845
		FF58		C6 D5 0011A	7\$: TSTL	ALLOCMAP	:	2848
				0F 13 0011E	BEQL	8\$:	
		FF58		C6 9F 00120	PUSHAB	ALLOCMAP	:	2851
		FF5C		C6 9F 00124	PUSHAB	ALLOCMAP_SIZE	:	
	67			02 FB 00128	CALLS	#2, LIB\$FREE_VM	:	
		FF58		C6 D4 0012B	CLRL	ALLOCMAP	:	2852
		FF68		C6 D5 0012F	8\$: TSTL	OLD_ALLOCMAP	:	2855
				11 13 00133	BEQL	9\$:	
		FF68		C6 9F 00135	PUSHAB	OLD_ALLOCMAP	:	2858
		00000000		EF 9F 00139	PUSHAB	P.AXI	:	
	67			02 FB 0013F	CALLS	#2, LIB\$FREE_VM	:	
		FF68		C6 D4 00142	CLRL	OLD_ALLOCMAP	:	2859
	50	FF6C		C6 3C 00146	9\$: MOVZWL	ERASE_CHANNEL, R0	:	2863
				0D 13 0014B	BEQL	10\$:	
				50 DD 0014D	PUSHL	R0	:	2866
		00000000G	00	01 FB 0014F	CALLS	#1, SYSSDASSGN	:	
				C6 B4 00156	CLRW	ERASE_CHANNEL	:	2867
				7E 7C 0015A	10\$: CLRQ	-(SP)	:	2871
				7E 7C 0015C	CLRQ	-(SP)	:	
				7E 7C 0015E	CLRQ	-(SP)	:	
				7E 7C 00160	CLRQ	-(SP)	:	
	7E			34 7D 00162	MOVQ	#52, -(SP)	:	

REBUILD
V04-000

E 11
16-Sep-1984 01:27:55 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:34 [MOUNT.SRC]REBUILD.B32;2

Page 83
(13)

REB
V04

	7E	FF6E	C6	3C	00165	MOVZWL	CHANNEL, -(SP)
			1A	DD	0016A	PUSHL	#26
	68		0C	FB	0016C	CALLS	#12, COMMON IO
		0C	A6	9F	0016F	PUSHAB	EXIT_HNDLR_DESC
00000000G	00		01	FB	00172	CALLS	#1, SYSSCANEXH
			7E	7C	00179	CLRD	-(SP)
00000000G	00		02	FB	0017B	CALLS	#2, SYSSUNWIND
	50		01	D0	00182	MOVL	#1, R0
			04	00185	11\$:	RET	

:
:
:
: 2876
:
: 2878
:
: 2881
: 2883

; Routine Size: 390 bytes, Routine Base: \$CODES + 14F4

```
2366 2884 1 ROUTINE RBLD_EXIT_HNDL : NOVALUE =
2367 2885 1
2368 2886 1 ++
2369 2887 1
2370 2888 1 Functional Description:
2371 2889 1
2372 2890 1 This routine is called by the OS on exit (for whatever reason) from
2373 2891 1 the DISKQUOTA utility. This routine must ensure that DISKQUOTA did
2374 2892 1 not leave things in an awkward state.
2375 2893 1
2376 2894 1 Calling Sequence:
2377 2895 1 standard
2378 2896 1
2379 2897 1 Input Parameters:
2380 2898 1 none
2381 2899 1
2382 2900 1 Implicit Inputs:
2383 2901 1 none
2384 2902 1
2385 2903 1 Output Parameters:
2386 2904 1 none
2387 2905 1
2388 2906 1 Implicit Outputs:
2389 2907 1 none
2390 2908 1
2391 2909 1 Routines Called:
2392 2910 1 none
2393 2911 1
2394 2912 1 Routine Value:
2395 2913 1 none
2396 2914 1
2397 2915 1 Signals:
2398 2916 1 none
2399 2917 1
2400 2918 1 Side Effects:
2401 2919 1 none
2402 2920 1
2403 2921 1 --
2404 2922 1
2405 2923 2 BEGIN
2406 2924 2
2407 2925 2
2408 2926 2 Make sure that DISKQUOTA did not leave a volume LOCKED.
2409 2927 2
2410 2928 2
2411 2929 2 IF .CLEANUP_FLAGS[CLF_UNLOCK]
2412 2930 2 THEN
2413 2931 3 BEGIN
2414 2932 3 CH$FILL (0, FIB$C_LENGTH, QUOTA_FIB);
2415 2933 3 QUOTA_FIB[FIB$W_CNTRLFUNC] = FIB$C_UNLK_VOL;
2416 2934 3 DO_IO (CHAN = .CHANNEL,
P 2935 3 FUNC = IOS_ACPCONTROL,
P 2936 3 P1 = QFIB_DESC
2417 2937 3 );
2418 2938 2 END;
2419 2939 2
2420 2940 1 END;
2421
2422 ! end of routine RBLD_EXIT_HNDL
```

				007C 00000 RBLD_EXIT HNDL:				
0040	8f	00	56 00000000'	EF 9E 00002	WORD	Save R2,R3,R4,R5,R6	: 2884	
			6E	00 2C 00009	MOVAB	QUOTA_FIB, R6	: 2932	
				66 00010	MOVCS	#0, (SP), #0, #64, QUOTA_FIB	: 2933	
		16	A6	08 B0 00011	MOVW	#8, QUOTA_FIB+22	: 2937	
				7E 7C 00015	CLRQ	-(SP)	: 2937	
				7E 7C 00017	CLRQ	-(SP)		
				7E D4 00019	CLRL	-(SP)		
			6C	A6 9F 0001B	PUSHAB	QFIB_DESC		
				7E 7C 0001E	CLRQ	-(SP)		
			7E	38 7D 00020	MOVQ	#56, -(SP)		
			7E	C6 3C 00023	MOVZWL	CHANNEL, -(SP)		
			00000000G 00	1A DD 00028	PUSHL	#26		
				0C FB 0002A	CALLS	#12, COMMON_IO	: 2940	
				04 00031	RET			

; Routine Size: 50 bytes, Routine Base: \$CODE\$ + 167A

```

2424 1 ROUTINE FILE_SIZE (HEADER) =
2425 1
2426 1 **
2427 1
2428 1 FUNCTIONAL DESCRIPTION:
2429 1
2430 1     This routine computes the number of blocks mapped by the specified
2431 1     file header.
2432 1
2433 1 CALLING SEQUENCE:
2434 1     FILE_SIZE (ARG1)
2435 1
2436 1 INPUT PARAMETERS:
2437 1     ARG1: header address
2438 1
2439 1 IMPLICIT INPUTS:
2440 1     NONE
2441 1
2442 1 OUTPUT PARAMETERS:
2443 1     NONE
2444 1
2445 1 IMPLICIT OUTPUTS:
2446 1     NONE
2447 1
2448 1 ROUTINE VALUE:
2449 1     number of blocks in header
2450 1
2451 1 SIDE EFFECTS:
2452 1     NONE
2453 1
2454 1 --
2455 1
2456 2 BEGIN
2457 2
2458 2 MAP
2459 2     HEADER          : REF BBLOCK;    ! file header arg
2460 2
2461 2 LINKAGE
2462 2     L_MAP_POINTER   = JSB :
2463 2                     GLOBAL (COUNT = 6, LBN = 7, MAP_POINTER = 8);
2464 2
2465 2 GLOBAL REGISTER
2466 2     COUNT           = 6,           ! retrieval pointer count
2467 2     LBN              = 7,           ! retrieval pointer LBN
2468 2     MAP_POINTER      = 8;           ! pointer to scan map area
2469 2
2470 2 LOCAL
2471 2     FILESIZE;                ! size of file
2472 2
2473 2 EXTERNAL ROUTINE
2474 2     GET_MAP_POINTER : L_MAP_POINTER; ! get value of file map pointer
2475 2
2476 2 ! Scan the map area. Count up the file size from the retrieval pointers.
2477 2 !
2478 2
2479 2 FILESIZE = 0;
2480 2 MAP_POINTER = .HEADER + .HEADER[FH2$B_MPOFFSET]*2;

```

```
: 2481      2998 2 UNTIL .MAP_POINTER GEQA .HEADER + (.HEADER[FH2$B_MPOFFSET] + .HEADER[FH2$B_MAP_INUSE]) * 2
: 2482      2999 DO
: 2483      3000 BEGIN
: 2484      3001 GET MAP POINTER ();
: 2485      3002 FILESIZE = .FILESIZE + .COUNT;
: 2486      3003 MARK_ALLOC (.COUNT, .LBN);
: 2487      3004 END;
: 2488      3005
: 2489      3006 RETURN .FILESIZE;
: 2490      3007
: 2491      3008 1 END;
```

! end of routine FILE_SIZE

.EXTRN GET_MAP_POINTER

OFFC 00000 FILE_SIZE:						
				.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	: 2941
				CLRL	FILESIZE	: 2996
52	04	AC	D0 00004	MOVL	HEADER, R2	: 2997
53	01	A2	9A 00008	MOVZBL	1(R2), R3	
58		6243	3E 0000C	MOVAW	(R2)[R3], MAP_POINTER	
50	3A	A2	9A 00010 1\$:	MOVZBL	58(R2), R0	: 2998
50		53	C0 00014	ADDL2	R3, R0	
50		6240	3E 00017	MOVAW	(R2)[R0], R0	
50		58	D1 0001B	CMPL	MAP_POINTER, R0	
		13	1E 0001E	BGEQU	2\$	
		00000000G	00 16 00020	JSB	GET MAP_POINTER	: 3001
54		56	C0 00026	ADDL2	COUNT, FILESIZE	: 3002
7E		56	7D 00029	MOVQ	COUNT, -(SP)	: 3003
FC88	CF	02	FB 0002C	CALLS	#2, MARK_ALLOC	
		DD	11 00031	BRB	1\$: 2998
50		54	D0 00033 2\$:	MOVL	FILESIZE, R0	: 3006
		04	00036	RET		: 3008

; Routine Size: 55 bytes, Routine Base: \$CODE\$ + 16AC

```
2493 3009 1 ROUTINE SET_FREE (RVN) : NOVALUE =
2494 3010 1
2495 3011 1 **
2496 3012 1
2497 3013 1 Functional Description:
2498 3014 1
2499 3015 1 This routine sets the number of free blocks in the volume control block.
2500 3016 1 This routine must be called in kernel mode.
2501 3017 1
2502 3018 1 Calling Sequence:
2503 3019 1 standard
2504 3020 1
2505 3021 1 Input Parameters:
2506 3022 1 none
2507 3023 1
2508 3024 1 Implicit Inputs:
2509 3025 1 none
2510 3026 1
2511 3027 1 Output Parameters:
2512 3028 1 none
2513 3029 1
2514 3030 1 Implicit Outputs:
2515 3031 1 none
2516 3032 1
2517 3033 1 Routines Called:
2518 3034 1 none
2519 3035 1
2520 3036 1 Routine Value:
2521 3037 1 none
2522 3038 1
2523 3039 1 Signals:
2524 3040 1 none
2525 3041 1
2526 3042 1 Side Effects:
2527 3043 1 none
2528 3044 1
2529 3045 1 --
2530 3046 1
2531 3047 2 BEGIN
2532 3048 2
2533 3049 2 LOCAL
2534 3050 2 RVT : REF BBLOCK, ! pointer to relative volume table
2535 3051 2 UCB : REF BBLOCK, ! pointer to volume UCB
2536 3052 2 VCB : REF BBLOCK; ! pointer to volume VCB
2537 3053 2
2538 3054 2 EXTERNAL ROUTINE
2539 3055 2 GET_CHANNELUCB; ! get UCB assigned to channel
2540 3056 2
2541 3057 2 UCB = GET_CHANNELUCB (.CHANNEL);
2542 3058 2 VCB = .UCB[UCB$-VCB];
2543 3059 2 RVT = .VCB[VCB$-RVT];
2544 3060 2 IF .RVT NEQ .UCB THEN UCB = .VECTOR[RVT[RVT$-UCBLST], .RVN - 1];
2545 3061 2 VCB = .UCB[UCB$-VCB];
2546 3062 2 VCB[VCB$-FREE] = .BLOCKS_AVAILABLE;
2547 3063 1 END; ! end of routine SET_FREE
```

.EXTRN GET_CHANNELUCB

				000C 00000 SET_FREE:			
	7E	00000000'	EF	3C	00002	.WORD	Save R2,R3
00000000G	00		01	FB	00009	MOVZWL	CHANNEL, -(SP)
	53	34	A0	D0	00010	CALLS	#1, GET_CHANNELUCB
	52	20	A3	D0	00014	MOVL	52(UCB), VCB
	50		52	D1	00018	MOVL	32(VCB), RVT
			09	13	0001B	CMP	RVT, UCB
	51	04	AC	D0	0001D	BEQ	1\$
	50	40	A241	D0	00021	MOVL	RVN, R1
	53	34	A0	D0	00026	MOVL	64(RVT)[R1], UCB
40	A3	00000000'	EF	D0	0002A	MOVL	52(UCB), VCB
			04	00032		MOVL	BLOCKS_AVAIL, 64(VCB)
						RET	

; Routine Size: 51 bytes, Routine Base: \$CODE\$ + 16E3

3009
3057
3058
3059
3060
3061
3062
3063

```

2549 3064 1 ROUTINE UPDATE_ALLOCMAP (RVN, ERASE_REQUESTED) =
2550 3065 1
2551 3066 1 ++
2552 3067 1
2553 3068 1 Functional Description:
2554 3069 1
2555 3070 1 This routine writes a new storage bitmap (BITMAP.SYS) to the
2556 3071 1 specified volume. Before writing each block, the existing
2557 3072 1 storage map is read and compared against the new map. If blocks
2558 3073 1 are being returned to the storage bitmap, they may be erased before
2559 3074 1 the bitmap is updated.
2560 3075 1
2561 3076 1 Calling Sequence:
2562 3077 1 standard
2563 3078 1
2564 3079 1 Input Parameters:
2565 3080 1 ARG1 : relative volume number (RVN) of this volume
2566 3081 1 ARG2 : boolean. Determines whether or not to erase blocks that
2567 3082 1 are returned to the available pool of blocks.
2568 3083 1
2569 3084 1 Implicit Inputs:
2570 3085 1 OWN storage defined at the beginning of this module.
2571 3086 1
2572 3087 1 Output Parameters:
2573 3088 1 none
2574 3089 1
2575 3090 1 Implicit Outputs:
2576 3091 1 CHANNEL : I/O channel to current volume
2577 3092 1
2578 3093 1 Routine Value:
2579 3094 1 1
2580 3095 1
2581 3096 1 Signals:
2582 3097 1 RBLDS_ACCBITMAP : BITMAP.SYS could not be opened or read.
2583 3098 1 RBLDS_WRTBITMAP : A write operation to BITMAP.SYS failed.
2584 3099 1 RBLDS_ERASEBLKS : Error erasing blocks returned to bitmap.
2585 3100 1
2586 3101 1 Side Effects:
2587 3102 1 Areas of the disk may be erased.
2588 3103 1
2589 3104 1 --
2590 3105 1
2591 3106 2 BEGIN ! Start of UPDATE_ALLOCMAP
2592 3107 2
2593 3108 2 LINKAGE
2594 3109 2 IOC_CONVERT = JSB (REGISTER=0, REGISTER=1, REGISTER=4, REGISTER=5;
2595 3110 2 REGISTER=1) :
2596 3111 2 PRESERVE (2,3,4,5)
2597 3112 2 NOTUSED (6,7,8,9,10,11);
2598 3113 2
2599 3114 2 BUILTIN
2600 3115 2 MOVPSL; ! VAX-11 instruction to fetch PSL
2601 3116 2
2602 3117 2 EXTERNAL ROUTINE
2603 3118 2 ERASE_BLOCKS : ADDRESSING_MODE (GENERAL),
2604 3119 2 GET_CHANNELUCB,
2605 3120 2 IOC$CVT_DEVNAM : IOC_CONVERT ADDRESSING_MODE (GENERAL),
```

```
: 2606      3121 2      LIB$FREE_VM      : ADDRESSING_MODE (GENERAL),
: 2607      3122 2      LIB$GET_VM      : ADDRESSING_MODE (GENERAL);
: 2608      3123 2
: 2609      3124 2 LITERAL
: 2610      3125 2      DEVNAM_SIZE      = 15,      ! max length of a device name
: 2611      3126 2      MAP_VBN          = 2;      ! first VBN of storage map
: 2612      3127 2
: 2613      3128 2 LOCAL
: 2614      3129 2      CURRENT_PSL      : BBLOCK [4],      ! current PSL
: 2615      3130 2      BIT_COUNT,      ! # of adjacent reclaimed clusters
: 2616      3131 2      BUFPTR          : REF BITVECTOR,      ! current buffer pointer for file
: 2617      3132 2      CURRENT_DEV      : BBLOCK [DSC$K'S_BLN], ! device name descriptor
: 2618      3133 2      DEVNAM_BUF       : BBLOCK [DEVNAM_SIZE], ! device name buffer
: 2619      3134 2      ERASE_STATUS,    ! holds erase operation status
: 2620      3135 2      RVT              : REF BBLOCK,      ! pointer to Relative Volume Table
: 2621      3136 2      STATUS,          ! saves local status values
: 2622      3137 2      UCB              : REF BBLOCK,      ! pointer to Unit Control Block
: 2623      3138 2      VBN,            ! current Virtual Block Number in file
: 2624      3139 2      VCB              : REF BBLOCK;      ! pointer to Volume Control Block
: 2625      3140 2
: 2626      3141 2
: 2627      3142 2      ! If this routine is being called from USER mode, then disable the
: 2628      3143 2      ! erase to prevent an ACCVIO when trying to examine the device data
: 2629      3144 2      ! base.
: 2630      3145 2
: 2631      3146 2      MOVPSL (CURRENT_PSL);      ! Fetch the current PSL
: 2632      3147 2      IF .CURRENT_PSL[PSL$V_CURMOD] EQL PSL$C_USER
: 2633      3148 2      THEN
: 2634      3149 2          ERASE_REQUESTED = 0;
: 2635      3150 2
: 2636      3151 2
: 2637      3152 2      ! Open the storage allocation bitmap file (BITMAP.SYS) on the volume.
: 2638      3153 2
: 2639      3154 2      QUOTA_FIB[FIB$L_ACCTL] = FIB$M_WRITE OR FIB$M_NOWRITE;
: 2640      3155 2      QUOTA_FIB[FIB$W_FID_NUM] = FID$C_BITMAP;
: 2641      3156 2      QUOTA_FIB[FIB$W_FID_SEQ] = FID$C_BITMAP;
: 2642      3157 2      QUOTA_FIB[FIB$W_FID_RVN] = .RVN;
: 2643      3158 2
: 2644      3159 2      STATUS = DO_IO (CHAN = .CHANNEL,
: 2645      3160 2          FUNC = IO$ ACCESS OR IO$M_ACCESS,
: 2646      3161 2          IOSB = IO_STATUS,
: 2647      3162 2          P1 = QFIB_DESC
: 2648      3163 2      );
: 2649      3164 2      IF .STATUS THEN STATUS = .IO_STATUS[0];
: 2650      3165 2      IF NOT .STATUS
: 2651      3166 2      THEN
: 2652      3167 2          RBLD_EXIT (RBLD$ACCBITMAP, .RVN, .STATUS);
: 2653      3168 2
: 2654      3169 2
: 2655      3170 2      ! For each block in the bitmap, read the old map block and compare
: 2656      3171 2      ! it to the map we have laboriously constructed. If any blocks are
: 2657      3172 2      ! being reclaimed (that is, they are marked free on the new map and
: 2658      3173 2      ! marked 'in use' on the old map) then erase them before writing the
: 2659      3174 2      ! new map block.
: 2660      3175 2
: 2661      3176 2      ! We do not read the old storage map if the caller did not request
: 2662      3177 2      ! that reclaimed blocks be erased. If the map read-buffer cannot
```

```
2663 3178 2 ! be created, the erase is not attempted.
2664 3179 2
2665 3180 2 Note: The SCB is VBN 1 of BITMAP.SYS. The actual storage allocation
2666 3181 2 bitmap begins at VBN 2.
2667 3182 2
2668 3183 2
2669 3184 2 ERASE_STATUS = 1;
2670 3185 2 OLD_ALLOCMAP = 0;
2671 3186 2
2672 3187 2 IF .ERASE_REQUESTED ! ... then allocate read-buffer
2673 3188 2 THEN
2674 3189 2 IF .OLD_ALLOCMAP EQL 0
2675 3190 2 THEN
2676 3191 2 IF NOT (ERASE_STATUS = LIB$GET_VM (UPLIT (512), OLD_ALLOCMAP))
2677 3192 2 THEN
2678 3193 2 ERASE_REQUESTED = 0; ! Disable the erase
2679 3194 2
2680 3195 2 IF .ERASE_REQUESTED ! ... then assign a channel
2681 3196 2 THEN
2682 3197 2 BEGIN
2683 3198 2
2684 3199 2 Assign a channel to current volume so the erase I/O will work properly.
2685 3200 2 This is done by getting the UCB address of the root volume of the volume
2686 3201 2 set, following the pointers to the RVT, and from there picking up the UCB
2687 3202 2 address of this volume. Given the proper UCB address, format the device
2688 3203 2 name and assign a channel to the device.
2689 3204 2
2690 3205 2 Note that CHANNEL is assigned to the root volume of the volume set.
2691 3206 2
2692 3207 2 IF .ERASE_CHANNEL NEQ 0
2693 3208 2 THEN
2694 3209 2 $DASSGN (CHAN=.ERASE_CHANNEL);
2695 3210 2 UCB = KERNEL CALL (GET_CHANNELUCB, .CHANNEL);
2696 3211 2 VCB = .UCB[UCB$SL_VCB];
2697 3212 2 RVT = .VCB[VCB$SL_RVT];
2698 3213 2 IF .RVT NEQ .UCB
2699 3214 2 THEN
2700 3215 2 UCB = .VECTOR[RVT[RVT$SL_UCBLST], .RVN-1];
2701 3216 2 CURRENT_DEV[DSC$A_POINTER] = DEVNAM BUF;
2702 3217 2 IOC$CVT_DEVNAM (DEVNAM SIZE, DEVNAM BUF, -1, .UCB; CURRENT_DEV[DSC$W_LENGTH]);
2703 3218 2 CURRENT_DEV[DSC$B_DTYPE] = DSC$K_DTYPE_T;
2704 3219 2 CURRENT_DEV[DSC$B_CLASS] = DSC$K_CLASS_S;
2705 3220 2 IF NOT (ERASE_STATUS = $ASSIGN (DEVNAM=CURRENT_DEV, CHAN=ERASE_CHANNEL))
2706 3221 2 THEN
2707 3222 2 ERASE_REQUESTED = 0;
2708 3223 2
2709 3224 2 END;
2710 3225 2 VBN = MAP_VBN; ! Skip the SCB
2711 3226 2 BUFPTR = .ALLOCMAP;
2712 3227 2 UNTIL .BUFPTR GEQA .ALLOCMAP + .ALLOCMAP_SIZE DO
2713 3228 2 BEGIN ! Start of DO-UNTIL loop
2714 3229 2 IF .ERASE_REQUESTED
2715 3230 2 THEN
2716 3231 2 BEGIN ! Start of scan/erase code
2717 3232 2
2718 3233 2 Read the next block of the old bitmap. Read errors will abort the
2719 3234 2 rebuild and leave the volume software writelocked.
```

```

: 2720      3235  4      !
: 2721      3236  4      STATUS = DO_IO (CHAN = .CHANNEL,
: 2722      3237  4      FUNC = IOS_READVBLK,
: 2723      3238  4      IOSB = IO_STATUS,
: 2724      3239  4      P1   = .OLD_ALLOCMAP,
: 2725      3240  4      P2   = $12,
: 2726      3241  4      P3   = .VBN
: 2727      3242  4      );
: 2728      3243  4      IF .STATUS
: 2729      3244  4      THEN
: 2730      3245  5          IF NOT (STATUS = .IO_STATUS[0])
: 2731      3246  4          THEN
: 2732      3247  5              BEGIN
: 2733      3248  5                  CLEANUP_FLAGS[CLF_UNLOCK] = 0;
: 2734      3249  5                  RBLD_EXIT (RBLD$_ACCBITMAP, .RVN, .STATUS);
: 2735      3250  4                  END;
: 2736      3251  4      !
: 2737      3252  4      ! Compare the old bitmap block against the new bitmap block.
: 2738      3253  4      ! Reclaimed blocks are those marked 'free' on the new map
: 2739      3254  4      ! and 'in use' on the old.
: 2740      3255  4      !
: 2741      3256  4      BIT_COUNT = 0;
: 2742      3257  4      INCR I FROM 0 TO 4095 DO
: 2743      3258  5          BEGIN                                ! Start of scan loop
: 2744      3259  5              IF .BUFPTRE[I] AND NOT .OLD_ALLOCMAP[I]
: 2745      3260  5              THEN
: 2746      3261  5                  !
: 2747      3262  5                  ! This cluster is being reclaimed. Count it.
: 2748      3263  5                  !
: 2749      3264  5                  BIT_COUNT = .BIT_COUNT + 1
: 2750      3265  5              ELSE
: 2751      3266  6                  IF (.BIT_COUNT NEQ 0)
: 2752      3267  5                  THEN
: 2753      3268  6                      BEGIN
: 2754      3269  6                          !
: 2755      3270  6                          ! The last BIT_COUNT clusters are being reclaimed.
: 2756      3271  6                          ! Calculate the starting LBN and number of blocks
: 2757      3272  6                          ! being reclaimed and erase them.
: 2758      3273  6                          !
: 2759      3274  6                          STATUS = ERASE_BLOCKS (((.VBN-MAP_VBN)*4096)+(.I-.BIT_COUNT)) * .CLUSTER_FACTOR[.RVN-1]
: 2760      3275  6                          .BIT_COUNT * .CLUSTER_FACTOR[.RVN-1],
: 2761      3276  6                          .ERASE_CHANNEL
: 2762      3277  6                          );
: 2763      3278  6                          IF NOT .STATUS AND .ERASE_STATUS
: 2764      3279  6                          THEN
: 2765      3280  6                              ERASE_STATUS = .STATUS;
: 2766      3281  6                              BIT_COUNT = 0;
: 2767      3282  5                              END;
: 2768      3283  4                      END;                                ! End of scan loop
: 2769      3284  3                  END;                                ! End of scan/erase code
: 2770      3285  3
: 2771      3286  3
: 2772      3287  3      ! Write the new storage allocation bitmap block to the disk. Write errors
: 2773      3288  3      ! will abort the rebuild and leave the volume software writelocked.
: 2774      3289  3
: 2775      3290  3      STATUS = DO_IO (CHAN = .CHANNEL,
: 2776      3291  3      FUNC = IOS_WRITEVBLK,
```

```

: 2777      P 3292      3      IOSB = IO STATUS,
: 2778      P 3293      3      P1  = .BUFPTR,
: 2779      P 3294      3      P2  = 512,
: 2780      P 3295      3      P3  = .VBN
: 2781      3296      3      );
: 2782      3297      3      IF .STATUS
: 2783      3298      3      THEN
: 2784      3299      4      IF NOT (STATUS = .IO_STATUS[0])
: 2785      3300      3      THEN
: 2786      3301      4      BEGIN
: 2787      3302      4      !      CLEANUP_FLAGS[CLF_UNLOCK] = 0;
: 2788      3303      4      RBLD_EXIT (RBLD$_ORTBITMAP, .RVN, .STATUS);
: 2789      3304      3      END;
: 2790      3305      3
: 2791      3306      3      VBN = .VBN + 1;      ! Advance to next VBN in file
: 2792      3307      3      BUFPTR = .BUFPTR + 512;      ! Advance to next block in new bitmap
: 2793      3308      2      END;      ! End of DO-UNTIL loop
: 2794      3309      2
: 2795      3310      2      !
: 2796      3311      2      ! If we get this far, the storage bitmap has been updated.
: 2797      3312      2      ! Report errors encountered while erasing reclaimed blocks, if any.
: 2798      3313      2
: 2799      3314      2      IF NOT .ERASE_STATUS
: 2800      3315      2      THEN
: 2801      3316      2      RBLD_MESSAGE (RBLD$_ERASEBLKS, .RVN, .ERASE_STATUS);
: 2802      3317      2
: 2803      3318      2      !
: 2804      3319      2      ! Return resources to system.
: 2805      3320      2
: 2806      3321      2      IF .OLD_ALLOCMAP NEQ 0      ! Return buffer
: 2807      3322      2      THEN
: 2808      3323      3      BEGIN
: 2809      3324      3      LIB$FREE_VM (UPLIT (512), OLD_ALLOCMAP);
: 2810      3325      3      OLD_ALLOCMAP = 0;
: 2811      3326      2      END;
: 2812      3327      2
: 2813      3328      2      IF .ERASE_CHANNEL NEQ 0      ! Dassign channel
: 2814      3329      2      THEN
: 2815      3330      3      BEGIN
: 2816      3331      3      $DASSGN (CHAN=.ERASE_CHANNEL);
: 2817      3332      3      ERASE_CHANNEL = 0;
: 2818      3333      2      END;
: 2819      3334      2
: 2820      3335      2      RETURN 1
: 2821      3336      2
: 2822      3337      1      END;      ! End of UPDATE_ALLOCMAP
```

.PSECT \$SPLITS,NOWRT,NOEXE,2

00000200 0001C P.AAJ: .LONG 512
00000200 00020 P.AAK: .LONG 512

.EXTRN ERASE_BLOCKS, IOC\$CVT_DEVNAM
.EXTRN SYSS\$ASSIGN

.PSECT \$CODE\$,NOWRT,2

OFFC 00000 UPDATE_ALLOCMAP:

03	50	5B 00000000G	00 9E 00002	.WORD	Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11	3064
		5A 00000000'	EF 9E 00009	MOVAB	COMMON_IO, R11	
		5E	18 C2 00010	MOVAB	OLD_ALLOCMAP, R10	
			50 DC 00013	SUBL2	#24, SP	
		02	18 ED 00015	MOVPSL	CURRENT_PSL	3146
			03 12 0001A	CMPZV	#24, #2, CURRENT_PSL, #3	3147
			08 AC D4 0001C	BNEQ	1\$	
		00D8 CA 0101	8F 3C 0001F 1\$:	CLRL	ERASE_REQUESTED	3149
		00DC CA 00020002	8F DO 00026	MOVZWL	#257, QUOTA_FIB	3154
		53 04	AC DO 0002F	MOVL	#131074, QUOTA_FIB+4	3155
		00E0 CA	53 B0 00033	MOVL	RVN, R3	3157
			7E 7C 00038	MOVW	R3, QUOTA_FIB+8	
			7E 7C 0003A	CLRQ	-(SP)	3163
			7E D4 0003C	CLRQ	-(SP)	
		0144	CA 9F 0003E	CLRL	-(SP)	
			7E 7C 00042	PUSHAB	QFIB_DESC	
		0C	AA 9F 00044	CLRQ	-(SP)	
		7E 72	8F 9A 00047	PUSHAB	IO STATUS	
		7E 06	AA 3C 0004B	MOVZBL	#1T4, -(SP)	
			1A DD 0004F	MOVZWL	CHANNEL, -(SP)	
		6B	0C FB 00051	PUSHL	#26	
		56	50 DO 00054	CALLS	#12, COMMON_IO	
		07	56 E9 00057	MOVL	R0, STATUS	
		56 0C	AA 3C 0005A	BLBC	STATUS, 2\$	3164
		11	56 E8 0005E	MOVZWL	IO STATUS, STATUS	
			8F BB 00061 2\$:	BLBS	STATUS, 3\$	3165
		0048	8F DD 00065	PUSHR	#*M<R3,R6>	3167
		00450048	03 FB 0006B	PUSHL	#4522056	
		00	01 DO 00072 3\$:	CALLS	#3, LIB\$STOP	
		59	6A D4 00075	MOVL	#1, ERASE STATUS	3184
		1A 08	AC E9 00077	CLRL	OLD_ALLOCMAP	3185
			18 12 0007B	B_BC	ERASE_REQUESTED, 4\$	3187
			5A DD 0007D	BNEQ	4\$	3189
			EF 9F 0007F	PUSHL	R10	3191
		00000000G	02 FB 00085	PUSHAB	P.AAJ	
		59	50 DO 0008C	CALLS	#2, LIB\$GET VM	
		03	59 E8 0008F	MOVL	R0, ERASE STATUS	
			AC D4 00092	BLBS	ERASE STATUS, 4\$	
		71 08	AC E9 00095 4\$:	CLRL	ERASE_REQUESTED	3193
		50 04	AA 3C 00099	BLBC	ERASE_REQUESTED, 7\$	3195
			09 13 0009D	MOVZWL	ERASE_CHANNEL, R0	3207
			50 DD 0009F	BEQL	5\$	
		00000000G	01 FB 000A1	PUSHL	R0	3209
		7E 06	AA 3C 000A8 5\$:	CALLS	#1, SYSSDASSGN	
			01 DD 000AC	MOVZWL	CHANNEL, -(SP)	3210
			5E DD 000AE	PUSHL	#1	
		00000000G	00 9F 000B0	PUSHL	SP	
		9F	04 FB 000B6	PUSHAB	GET_CHANNELUCB	
		52	50 DO 000BD	CALLS	#4, @SYSSCMKRN	
		50 34	A2 DO 000C0	MOVL	R0, UCB	3211
		50 20	A0 DO 000C4	MOVL	52(UCB), VCB	3212
		52	50 D1 000C8	MOVL	32(VCB), RVT	3213
			05 13 000CB	CPL	RVT, UCB	
				BEQL	6\$	

14	52	40	A043	DO	000CD	MOVL	64(RVT)[R3], UCB	3215
	AE		6E	9E	000D2	MOVAB	DEVNAM_BUF, CURRENT_DEV+4	3216
	51		6E	9E	000D6	MOVAB	DEVNAM_BUF, R1	3217
	55		52	DO	000D9	MOVL	UCB, R5	
	54		01	CE	000DC	MNEGL	#1, R4	
	50		0F	DO	000DF	MOVL	#15, R0	
		0C000000G	00	16	000E2	JSB	IOC\$CVT DEVNAM	
10	AE		51	BO	000E8	MOVW	R1, CURRENT_DEV	
12	AE	010E	8F	BO	000EC	MOVW	#270, CURRENT_DEV+2	3218
			7E	7C	000F2	CLRQ	-(SP)	3220
		04	AA	9F	000F4	PUSHAB	ERASE CHANNEL	
		1C	AE	9F	000F7	PUSHAB	CURRENT_DEV	
00000000G	00		04	FB	000FA	CALLS	#4, SYSSASSIGN	
	59		50	DO	00101	MOVL	R0, ERASE_STATUS	
	03		59	E8	00104	BLBS	ERASE_STATUS, 7\$	
		08	AC	D4	00107	CLRL	ERASE-REQUESTED	3222
	55		02	DO	0010A	MOVL	#2, VBN	3225
	58	F0	AA	DO	0010D	MOVL	ALLOCMAP, BUFPTR	3226
50	AA	F4	AA	C1	00111	ADDL3	ALLOCMAP_SIZE, ALLOCMAP, R0	3227
	50		58	D1	00117	CMPL	BUFPTR, R0	
			03	1F	0011A	BLSSU	9\$	
			00E1	31	0011C	BRW	18\$	
	03	08	AC	E8	0011F	BLBS	ERASE-REQUESTED, 10\$	3229
			0095	31	00123	BRW	16\$	
			7E	7C	00126	CLRQ	-(SP)	3242
			7E	D4	00128	CLRL	-(SP)	
			55	DD	0012A	PUSHL	VBN	
	7E	0200	8F	3C	0012C	MOVZWL	#512, -(SP)	
			6A	DD	00131	PUSHL	OLD_ALLOCMAP	
			7E	7C	00133	CLRQ	-(SP)	
		0C	AA	9F	00135	PUSHAB	IO STATUS	
			31	DD	00138	PUSHL	#49	
	7E	06	AA	3C	0013A	MOVZWL	CHANNEL, -(SP)	
			1A	DD	0013E	PUSHL	#26	
	68		0C	FB	00140	CALLS	#12, COMMON_IO	
	56		50	DO	00143	MOVL	R0, STATUS	
	18		56	E9	00146	BLBC	STATUS, 11\$	3243
	56	0C	AA	3C	00149	MOVZWL	IO STATUS, STATUS	3245
	11		56	E8	0014D	BLBS	STATUS, 11\$	
		0048	8F	BB	00150	PUSHR	#*M<R3,R6>	3249
		00450048	8F	DD	00154	PUSHL	#4522056	
00000000G	00		03	FB	0015A	CALLS	#3, LIB\$STOP	
			57	D4	00161	CLRL	BIT_COUNT	3256
			54	D4	00163	CLRL	I	3257
09	68		54	E1	00165	BBC	I, (BUFPTR), 13\$	3259
04	00	BA	54	E0	00169	BBS	I, OLD_ALLOCMAP, 13\$	
			57	D6	0016E	INCL	BIT_COUNT	3264
			41	11	00170	BRB	15\$	
			57	D5	00172	TSTL	BIT_COUNT	3266
			3D	13	00174	BEQL	15\$	
	7E	04	AA	3C	00176	MOVZWL	ERASE CHANNEL, -(SP)	3276
	52	0120	DA43	3E	0017A	MOVW	@CLUSTER_FACTOR[R3], R2	3275
	50	FE	A2	3C	00180	MOVZWL	-2(R2), R0	
7E	57		50	C5	00184	MULL3	R0, BIT_COUNT, -(SP)	
50	55		0C	78	00188	ASHL	#12, VBN, R0	3274
51	54		57	C3	0018C	SUBL3	BIT_COUNT, I, R1	
	50	E000	C140	9E	00190	MOVAB	-8192(R1)[R0], R0	

7E	00000000G	51	FE	A2	3C	00196	MOVZWL	-2(R2), R1		
		50		51	C5	0019A	MULL3	R1, R0, -(SP)		
		00		03	FB	0019E	CALLS	#3, ERASE_BLOCKS		
		56		50	D0	001A5	MOVL	R0, STATUS		
		06		56	E8	001A8	BLBS	STATUS, 14\$		3278
		03		59	E9	001AB	BLBC	ERASE STATUS, 14\$		
		59		56	D0	001AE	MOVL	STATUS, ERASE_STATUS		3280
AA		54	00000FFF	57	D4	001B1	14\$: CLR	BIT COUNT		3281
				8F	F3	001B3	15\$: AOBLEQ	#4095, 1, 12\$		3257
				7E	7C	001BB	16\$: CLRQ	-(SP)		3296
				7E	D4	001BD	CLR	-(SP)		
		7E	0200	55	DD	001BF	PUSHL	VBN		
				8F	3C	001C1	MOVZWL	#512, -(SP)		
				58	DD	001C6	PUSHL	BUFPTR		
				7E	7C	001C8	CLRQ	-(SP)		
			0C	AA	9F	001CA	PUSHAB	IO STATUS		
				30	DD	001CD	PUSHL	#48		
		7E	06	AA	3C	001CF	MOVZWL	CHANNEL, -(SP)		
				1A	DD	001D3	PUSHL	#26		
		68		0C	FB	001D5	CALLS	#12, COMMON_IO		
		56		50	D0	001D8	MOVL	R0, STATUS		
		18		56	E9	001DB	BLBC	STATUS, 17\$		3297
		56	0C	AA	3C	001DE	MOVZWL	IO STATUS, STATUS		3299
		11		56	E8	001E2	BLBS	STATUS, 17\$		
			0048	8F	BB	001E5	PUSHR	#*M<R3,R6>		3303
			00450080	8F	DD	001E9	PUSHL	#4522112		
		00		03	FB	001EF	CALLS	#3, LIB\$STOP		
				55	D6	001F6	17\$: INCL	VBN		3306
		58	0200	C8	9E	001F8	MOVAB	512(R8), BUFPTR		3307
				FF	31	001FD	BRW	8\$		3227
		11		59	E8	00200	18\$: BLBS	ERASE STATUS, 19\$		3314
			0208	8F	BB	00203	PUSHR	#*M<R3,R9>		3316
			004500B8	8F	DD	00207	PUSHL	#4522168		
		00		03	FB	0020D	CALLS	#3, LIB\$SIGNAL		
				6A	D5	00214	19\$: TSTL	OLD_ALLOCMAP		3321
				11	13	00216	BEQL	20\$		
				5A	DD	00218	PUSHL	R10		3324
			00000000'	EF	9F	0021A	PUSHAB	P.AAK		
		00		02	FB	00220	CALLS	#2, LIB\$FREE_VM		
				6A	D4	00227	CLR	OLD_ALLOCMAP		3325
		50	04	AA	3C	00229	20\$: MOVZWL	ERASE_CHANNEL, R0		3328
				0C	13	0022D	BEQL	21\$		
				50	DD	0022F	PUSHL	R0		3331
		00		01	FB	00231	CALLS	#1, SYSSDASSGN		
			04	AA	B4	00238	CLR	ERASE_CHANNEL		3332
		50		01	D0	0023B	21\$: MOVL	#1, R0		3335
				04	0023E		RET			3337

; Routine Size: 575 bytes, Routine Base: \$CODE\$ + 1716

: 2823 3338 1
: 2824 3339 1 end
: 2825 3340 1
: 2826 3341 0 eludom

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes						
MSG_TEXT	1559	NOVEC,NOWRT,	RD	NOEXE,NOSHR,	LCL,	REL,	CON,NOPICT,ALIGN(0)	
MSG_INDEX	96	NOVEC,NOWRT,	RD	NOEXE,NOSHR,	LCL,	REL,	CON,NOPICT,ALIGN(2)	
SOWNS	424	NOVEC, WRT,	RD	NOEXE,NOSHR,	LCL,	REL,	CON,NOPICT,ALIGN(2)	
SCODES	6485	NOVEC,NOWRT,	RD	EXE,NOSHR,	LCL,	REL,	CON,NOPICT,ALIGN(2)	
SPLITS	36	NOVEC,NOWRT,	RD	NOEXE,NOSHR,	LCL,	REL,	CON,NOPICT,ALIGN(2)	

Library Statistics

File	----- Symbols -----		Pages Mapped	Processing Time
	Total	Loaded Percent		
_S255SDUA28:[SYSLIB]LIB.L32;1	18619	126 0	1000	00:01.9

: Information: 1
: Warnings: 0
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:REBUILD/OBJ=OBJ\$:REBUILD MSRC\$:REBUILD/UPDATE=(ENH\$:REBUILD)

: Size: 6485 code + 2115 data bytes
: Run Time: 01:47.0
: Elapsed Time: 03:00.7
: Lines/CPU Min: 1874
: Lexemes/CPU-Min: 17843
: Memory Used: 833 pages
: Compilation Complete

0246 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

